Nr. 7/85 Juli
DM 6,50, sfr 6,50, öS 50, Lit 5900, hfl 7,50

PEEKER

MAGAZIN FÜR APPLE-COMPUTER

Pyramid Pitty

AP33-RAM-Karte

Formatter

6502 leicht gemacht

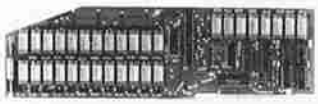
Wordstar und Epson

Megacore-Festplatte

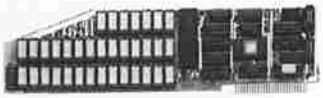
Zeichensatz-EPROM

Pascal-Preisausschreiben

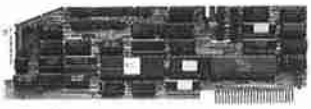
Hüthig PUBLIKATION



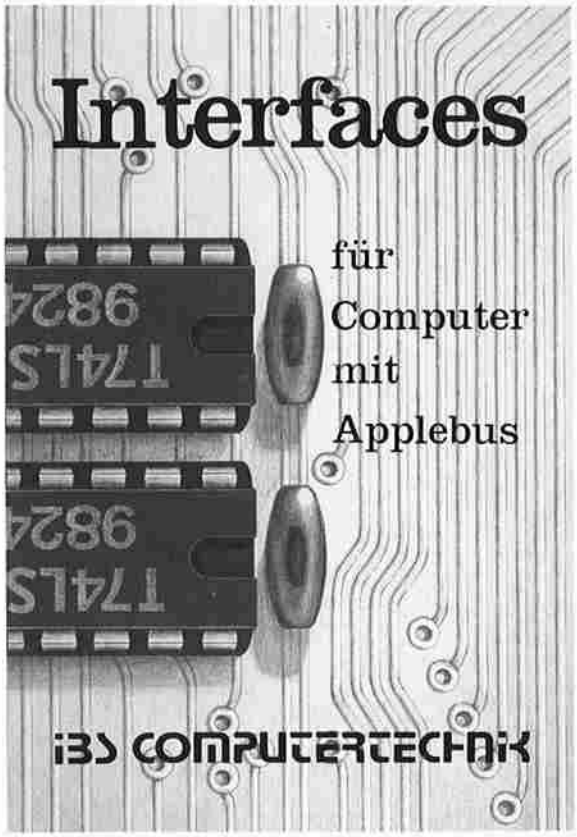
AP 13 und AP 17
RAM-Karten zum Einsatz als Pseudodisk unter CP/M, USCD und APPLE-DOS. Speichergröße von 64 kByte bis 256 kByte.
Bestell-Nr.: A 1013 a-b
A 1017 a-d



AP 33
RAMDISK der neuen Generation. Für besonders speicherintensive Arbeiten ist der Ausbau in Stufen von 64 kByte bis 1MByte möglich.
Bestell-Nr. A 1033



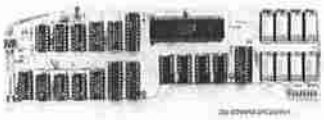
AP 14
Floppy-Controller für alle Anwendungsfälle. 10 Laufwerke können gleichzeitig angeschlossen werden. 4 × 8" DSDD, 4 × 5¼" DSDD und zwei Apple-Standardlaufwerke. Maximal ca. 10MByte im Direktzugriff.
Bestell-Nr.: A 1014



AP 19
12-Kanal AD-DA-Wandler mit 12 bit Auflösung und 25 μ sec Wandlungszeit. Eingangsspannung ±10 V. Ein schneller Wandler für extrem schnelle Anwendungen.
Bestell-Nr.: A 1019

NEU! jetzt 512 k-RAM

AP 20
INTEMEX mit 68 000 CPU und 128 k-RAM. Diese Karte macht aus Ihrem Rechner mit „Applebus" einen echten 16 bit-Rechner. Eine Zusatzkarte (AP 26) ermöglicht einen Arbeitsspeicher bis zu einem MByte und an Software gibt es einiges. Z.B. stehen drei Betriebssysteme und die wichtigsten Hochsprachen zur Verfügung.
Bestell-Nr. A 1020



NEU! 8 MHz Takt

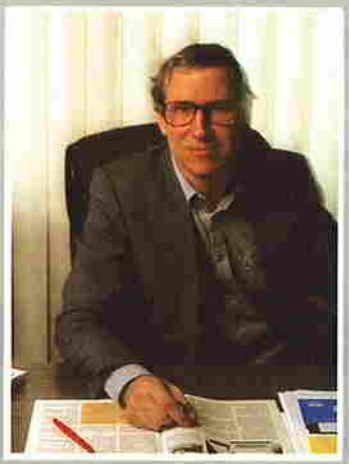
AP 22
INTEMEX mit Z 80 B-CPU und 64 k-RAM. Wenn Sie einmal diese Karte in Aktion gesehen haben, werden Sie auch feststellen: „Geschwindigkeit ist keine Hexerei, man braucht nur die AP 22". Mit dieser Karte wird Ihr APPLE II zum z.Z. schnellsten CP/M-Computer, und in Verbindung mit dem SPACE 84 erhalten Sie Computerleistung, die wirklich einmalig ist. Wir vermitteln gerne eine Vorführung.
Bestell-Nr. A 1022

NEU!!!

Das Interface-Buch von IBS, ein Buch für Alle, die Ihren APPLE II oder Kompatiblen optimal nutzen wollen. Detaillierte Schaltpläne, Bauteilelisten und Benutzungshinweise zu allen IBS-Interfaces finden Sie jetzt in einem Buch vereint. Ausführliche Abhandlungen über Spezialschaltungen, über Anwendungsmöglichkeiten, über neue Softwarewelten aber auch über die Grenzen des APPLE II-Systems bestimmen den Wert dieses Buches.
Für nur DM 8,00 erhalten Sie dieses Buch ab sofort bei Ihrem Computerfachhändler oder für DM 8,00 + DM 2,00 Versandkosten bei IBS COMPUTERVERTRIEB.

IBS COMPUTERTECHNIK

Olper Straße 10 · 4800 Bielefeld 14 · Tel.: 0521/444032 · W. Germany
1011 Rose Marie Lane 16 · Stockton CA 95207 · Tel. 209/473-7473 USA



editorial

Wenn Sie dieses Heft in Händen halten, verbringe ich gerade meinen Urlaub in südlichen Gefilden. Wie das so üblich ist, nehme ich mir auch dieses Mal wieder amerikanische Fachliteratur mit, um mich auf dem laufenden zu halten...

...beispielsweise über Pascal, denn ich erwarte natürlich, daß viele an unserem Pascal-Wettbewerb teilnehmen werden, der in diesem Heft ausgeschrieben wird. Beim letzten Peeker-Wettbewerb über Primzahlen hatten einige Schulklassen teilgenommen, und ich würde mich freuen, wenn auch bei dem neuerlichen Preisausschreiben wieder einige Klassen ihren „Punktesaldo" einsenden würden. Leider mußte ich die Kürze des Pascal-Quellcodes in den „Punktesaldo" des Wettbewerbs mit einbeziehen, um „unerlaubte Tricks" zumindest einzuschränken. Dies wird zur Folge haben, daß völlig „undidaktisch" aussehende Listings erstellt werden, die nur entfernt an die ästhetische, übersichtliche Strukturierung eines typischen Pascal-Quelltextes erinnern werden. Informatik-Lehrer mögen hier einmal ein Auge zudrücken.

Auch wenn für die eigentlichen Fachbeiträge des Peeker das von der Mehrzahl der Leser gewünschte hohe Niveau beibehalten werden soll, wird mit diesem Heft eine Serie von meist 16seitigen, in sich geschlossenen Sonderartikeln beginnen, die auf Anfänger abgestimmt sind. Neben dem ersten Sonderteil „6502 leicht gemacht" in diesem Heft sind weitere Übersichtsartikel über 68000, Z80, MBASIC, Pascal und andere Themen in Arbeit. Bei den großen Spezialserien (Mac-BASIC, ProDOS, Graf-quattro) ist eine Aufsplittung auf mehrere Hefte aus Platzgründen unvermeidlich. Ich glaube jedoch, daß gerade dem Anfänger ein vollständiger Sonderteil in der Form eines Mini-Buches mehr bringt als eine auf mehrere Hefte verteilte Serie. Trotzdem würde mich gerne Ihre Meinung hierzu interessieren.

Ulrich Stiehl

Hüthig
PUBLIKATION

INHALT

7/85

Impressum
Peeker
Magazin für Apple-Computer
2. Jahrgang 1985
ISSN 0176-9200
© für den gesamten Inhalt einschließlich der Programme Dr. Alfred Hüthig Verlag, Heidelberg 1985

Verleger und Herausgeber:
Dipl.-Kfm. Holger Hüthig
Geschäftsführung Zeitschriften:
Heinz Melcher
Chefredakteur:
Ulrich Stiehl (us) Tel. (0 62 21) 48 93 52
(Bitte nur in redaktionellen Angelegenheiten anrufen)

Anzeigenleitung:
Jürgen Maurer, Tel. (0 62 21) 48 92 18
z. Zt. gilt Anzeigenpreisliste Nr. 3
Vertriebsleitung:
Ruth Biller, Tel. (0 62 21) 48 92 80
Produktionsleitung: Gunter Sokollek
Gestaltung: Rainer Schmitt
Titelbild: Creative Computer Service, Mannheim

PEEKER

MAGAZIN FÜR APPLE-COMPUTER

5 IMPRESSUM

HOBBY

6 **Pyramid Pitty**
Ein Reaktionsspiel
von Michael Matzat

TECHNIK

8 **Die AP33-Megawrap-RAM-Karte**
Mit einem RAM-Disk-Driver für ProDOS
von Ulrich Stiehl

20 **Formatter**
Ein universelles Formatierungsprogramm
von Arne Schäpers

29 **Bit Editor**
Zeichensatz-EPROMs für die Videx-Karte
von Joachim Klamt

SONDERTEIL

33 **6502 leicht gemacht**
von Ulrich Stiehl

CP/M

57 **Wordstar mit allen FX-80-Schriftarten**
von Dipl.-Ing. H. A. Rohrbacher

PASCAL

62 **Pascal-Preisausschreiben**

KURZBERICHTE

67 **No Orchids for Miss Lisa**
Lisa und die Folgen
von Ulrich Stiehl

ASSEMBLER

69 **Hex-Dez-Konvertierung für 32-Bit-Zahlen**
von Harald Grumser

71 **Vorlesestunde**
Apple und SAM – ein hilfreiches Gespann
von Dr. Jürgen B. Kehrel

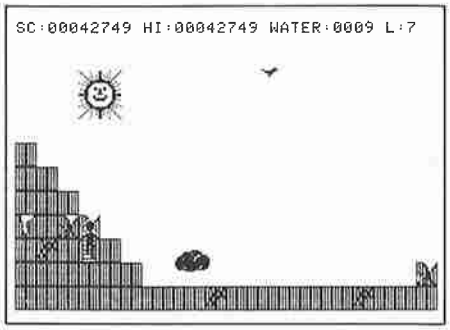
73 LESERBRIEFE

TESTBERICHTE

76 **MEGACORE**
Festplatte mit 10 Megabytes
getestet von Harald Grumser

77 **Star Delta-10 und Grafstar-Interface**
getestet von Karl-Walter Bott

78 INSERENTENVERZEICHNIS



Verlag:
Dr. Alfred Hüthig Verlag GmbH
Im Weiher 10, Postfach 10 28 69
6900 Heidelberg
Telefon (0 62 21) 4 89-0
Telex 4-6 17 27 hued d.

Erscheinungsweise: 12 Hefte jährlich,
Erscheinungstag jeweils 1 Woche vor Monatsbeginn.
Jahresabonnement DM 72,–, einschließlich MwSt,
im Inland portofrei. Einzelheft DM 6,50
Vertrieb Handel:
MZV – Moderner Zeitschriften Vertrieb GmbH
Breslauer Str. 5, Postfach 1123,
8057 Eching b. München,
Tel. 089/319 10 67, Telex 0 522 656

Zahlungen: an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: Postscheckkonten: BRD: Karlsruhe 485 45-753; Österreich: Wien 755588 88; Schweiz: Basel 40-24417; Niederlande: Den Haag 1 457 28; Italien: Mailand 47718; Belgien: Brüssel 723026; Dänemark: Kopenhagen 349 69; Norwegen: Oslo 994 24; Schweden: Stockholm 5477 76-5

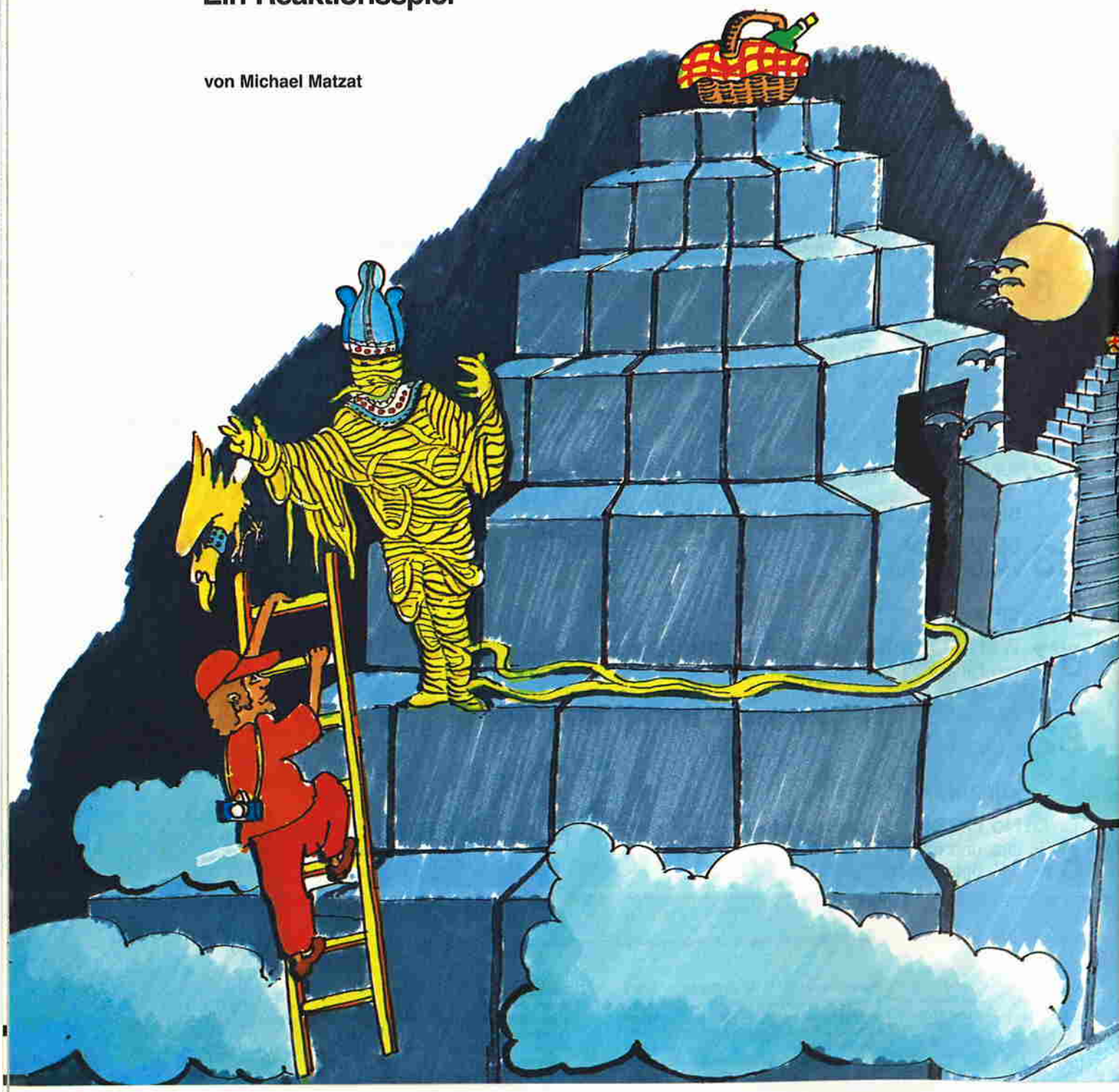
Bankkonten: Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 0 2165 041; BLZ 672 700 03; Bezirkssparkasse Heidelberg 204 51, BLZ 672 500 20.

Herstellung: Heidelberger Verlagsanstalt
Printed in Germany

Pyramid Pitty

Ein Reaktionsspiel

von Michael Matzat



Vorbemerkung
Für Liebhaber von Computerspielen bringen wir erstmals im Peeker ein umfangreiches und, wie wir meinen, erstaunlich professionelles Hires-Spiel. Erstaunlich deshalb, weil der Autor, Michael Matzat, obwohl er erst 19 Jahre alt ist, ein lupenreines Assemblerprogramm geschaffen hat, das hinsichtlich Schnelligkeit, Vielfalt und Komfort den meist oberhalb der 100-Mark-Grenze angesiedelten, kommerziellen Apple-Spielen kaum nachsteht. Wegen des Umfangs – mehrere tausend Zeilen Quellcode – ist es uns leider nicht möglich, das Programm abzudrucken. Es befindet sich als Objekt- und Quellcode auf der Peeker-Sammeldisk . us

Spielidee
Peter Pause, oder auch einfach nur Pitt genannt, kommt als einfacher Tourist nach Ägypten, wo er, wie soll es auch anders sein, natürlich eine der Pyramiden besucht, um auf ihr ein kleines Picknick einzunehmen. Als er wieder am Fuße der Pyramide steht, bemerkt er, daß er seinen Picknickkorb auf der Pyramide verloren hat.
Seine Aufgabe ist es nun, die verlorenen Picknicksachen wieder einzusammeln. Erschwerend kommt hinzu, daß die Pyramide schon sehr brüchig ist, man leicht stolpern und von der Pyramide stürzen kann. Doch das ist noch nicht alles! Aus einfachen Rissen in den Stufen der Pyramide können, ohne daß es Pitt bemerkt, tiefe Spalten werden.

Spielablauf
Das gesamte Spielfeld enthält eine kleine und eine große Pyramide und besteht aus 22 Hires-Teilbildern. Um von einem Teilbild in ein anderes zu gelangen, braucht man nur durch einen der 4 Bildschirmränder zu laufen. Sollte dies nicht möglich sein, so folgt in dieser Richtung kein weiteres Teilbild mehr.
Der Computer spielt mit mehreren Gegnern gegen Sie. Zunächst jeweils nur mit einem, aber in den höheren Leveln können auch bis zu drei Gegner gegen Sie antreten.
Ihr erster Gegner ist der *Pharao*. Er ist eigentlich ungefährlich, doch trotzdem sollte man es vermeiden, mit ihm auf der gleichen oder der nächst höheren Pyramidenstufe zu stehen.
Der *Falke* des Pharaos ist Ihr nächster Gegner. Achten Sie darauf, daß er Sie nicht an den Beinen berührt, da Sie sonst stolpern könnten.
Die *Wolke* versucht Sie einzuhüllen und Ihnen somit den Sichtkontakt mit dem Boden zu nehmen.
Am meisten macht Ihnen allerdings die Hitze zu schaffen. Durch sie verdunstet Ihr Wasser, und je höher der Level, desto schneller verdunstet es. Ihren Wasserverlust müssen Sie unbedingt auszugleichen versuchen. In den Löchern, die in fast jedem Teilbild enthalten sind, kann sich Wasser befinden. Oder auch nicht! Denn je höher der Level, desto seltener findet man gefüllte Wasserlöcher. Wenn ein Loch kein Wasser enthält, dann kann es sein, daß .... Aber dies soll hier nicht verraten werden.

Bewertung
Kommen wir nun zur Bewertung. Für jeden wiedergefundenen Gegenstand gibt es zwischen 1 und 1000 Punkte. Darüber hinaus erhalten Sie pro Liter gefundenen Wassers je einen Punkt und am Ende eines Levels bekommen Sie pro Liter verbliebenen Wassers jeweils 100 Punkte. Diese Angaben sind natürlich wie immer ohne Gewähr!

Bedienung
Das Spiel wird mit
RUN PYRAMID.PITTY
gestartet. In diesem vorgeschalteten Startprogramm können Sie dem eigentlichen Programm mitteilen, ob Sie über alle Cursor-Tasten verfügen oder nicht. Denn gespielt wird über Tastatur, wobei auf dem Apple IIe und IIc die Cursor-Tasten und auf dem Apple II Plus neben den Links- und Rechtspfeil-Tasten die Buchstaben A und Z die Hoch- und Tiefpfeiltasten ersetzen. Mit ESC kann das Spiel gestoppt und mit der Leertaste wieder gestartet werden. Wenn Sie die mitlaufende Melodie stört, so können Sie den Lautsprecher mit Ctrl-S abschalten, da Ctrl-S den Ton auf den Kassettenausgang legt.

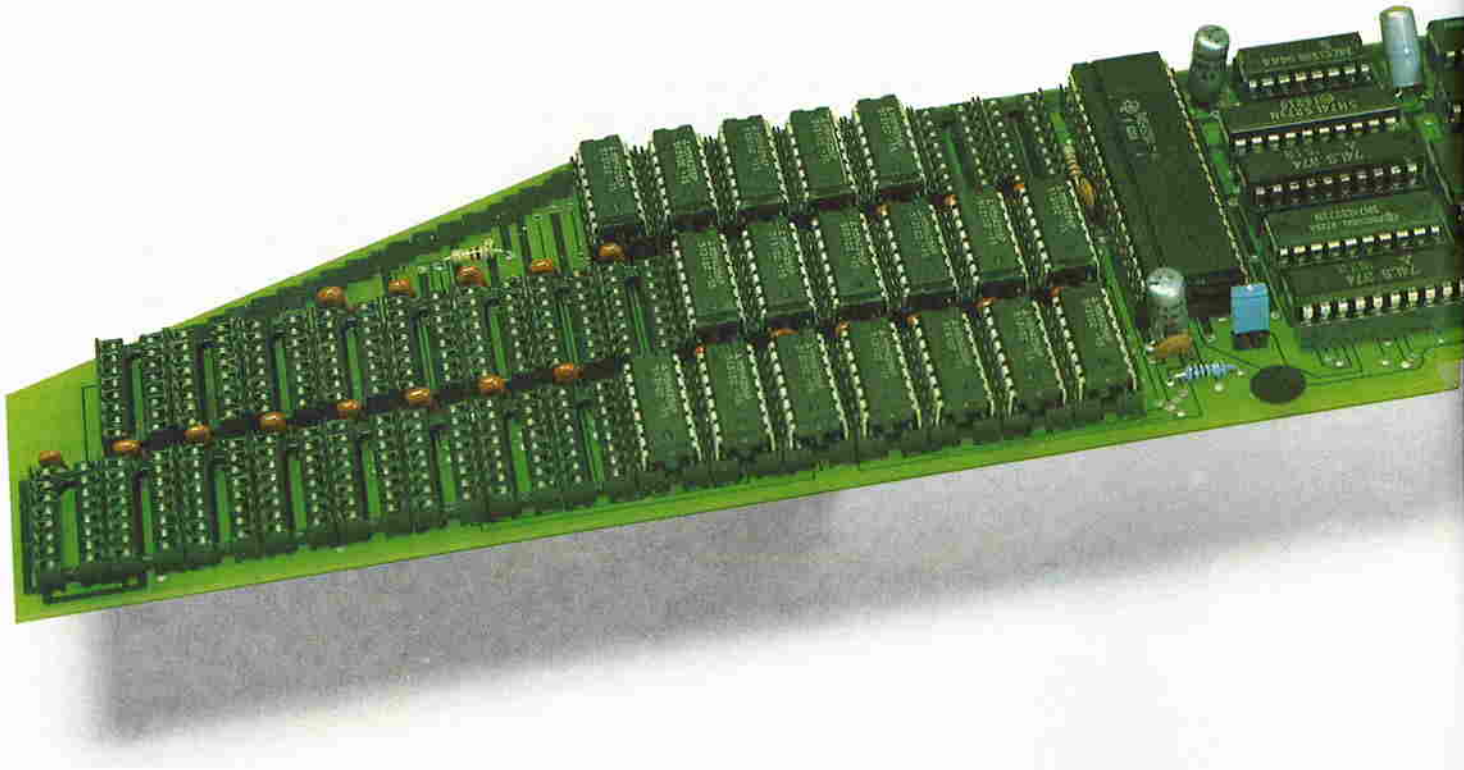
Technisches
Das Spiel ist komplett in Assembler geschrieben und kann trotz seines geraumen Umfangs (ca. 27K) noch erweitert werden, beispielsweise mit einer Tabelle der 10 besten Spieler des Tages usw. Aus Gründen der Geschwindigkeit, die ja bei Spielen sehr wichtig ist, war es leider nicht möglich, die normalen Möglichkeiten des Apples zur Darstellung von Grafik, wie etwa die DRAW-, XDRAW- und HPLOT-Funktionen zu benutzen. Dies war auch der Grund, warum ich auf die in professionellen Spielen üblichen Blockshapes – auch Sprites, Raster- oder Bitmustergrafiken genannt –, deren großer Vorteil ihre schnelle Darstellungsgeschwindigkeit ist, zurückgegriffen habe. Das Programm enthält daher diverse Routinen zum Arbeiten mit Blockshapes, die man leicht zu einem ganzen Grafiksystem ausbauen könnte. Die daraus resultierende Verarbeitungsgeschwindigkeit hat es mir ermöglicht, die oben erwähnte Begleitmelodie einzubauen, ohne daß sich dadurch die in solchen Fällen üblichen, ruckartigen Bewegungen einstellen.

Probieren Sie einmal aus, welchen Spielstand Sie bei Pyramid Pitty erreichen! Bisher konnte ich, obwohl ich alle Kniffe und Tricks in diesem Spiel kenne, „nur" bis zum Level 7 vorzudringen.

Die AP33-Megawarp-RAM-Karte

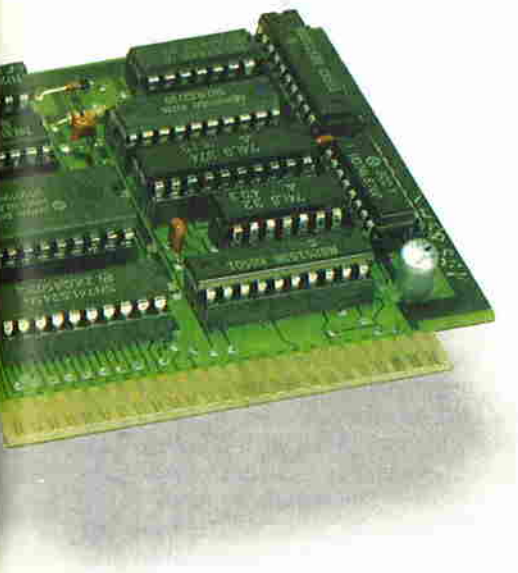
Mit einem RAM-Disk-Driver für ProDOS

von Ulrich Stiehl



Die neue RAM-Karte AP33 der Firma IBS in Bielefeld – auch als Megawarp bezeichnet – kann wahlweise mit 64-KBit- oder 256-KBit-RAM-Bausteinen bestückt werden. Am besten geht man gleich „in die vollen" und bestellt die Karte mit den neueren Chips, denn man kann 4 verschiedene Bestückungsstufen wählen, so daß der Geldbeutel nicht allzu stark strapaziert wird:

256K (ca. DM 1425,-)
512K (ca. DM 2160,-)
768K (ca. DM ????)
1024K (ca. DM 3625,-)
Ein einzelner 256-KBit-Chip oder IC (= Integrated Circuit) hat übrigens eine Speicherkapazität von 32K. Wenn von einem xyz-KBit-Chip oder xyz-K-IC die Rede ist, so teilen Sie xyz durch 8 und erhalten damit die Speicherkapazität in Kilobytes, also
256KBit : 8 = 32K, z.B. AP33
64KBit : 8 = 8K, z.B. IIe
16KBit : 8 = 2K, z.B. II+
Die voll bestückte AP33 hat damit 32 256KBit-RAM-Bausteine (32 * 32 = 1024K) sowie weitere CMOS-RAMs für den Slot-RAM-Bereich (s.u.).



1. AP33-Softswitches

Die mir zu Testzwecken zur Verfügung gestellte Karte (s. **Bild**) war zur Hälfte bestückt (512K). Im Gegensatz zu den früheren Adressierungsverfahren (s. Peeker, Heft 1/1985, S. 18) hat man bei der AP33 neue Wege beschritten. Früher wurden größere RAM-Karten meistens in einen Adreßbereich der „unteren" 64K „gemappt", z.B. in den Bereich $D000-$FFFF. Dieses Konzept ließ man nunmehr fallen. Statt dessen gibt es jetzt ein 20-Bit-Adreßregister (20 Bits, weil 2 hoch 20 = 1.048.576 = 1M = maximale Speicherkapazität), das eine lineare Adressierung von $000000 bis $0FFFFF (= 0 bis 1.048.576) erlaubt. Die 20 Bits sind auf 2 volle 8-Bit-Register sowie die eine Hälfte (= 4 Bits) eines weiteren 8-Bit-Registers verteilt, dessen andere Hälfte der Paritätsprüfung dient. Daneben gibt es noch ein 8-Bit-Datenregister, das jeweils das zu übertragende Byte aufnimmt.

Poke-Beispiel: Nehmen wir an, wir wollen das Byte $CC in die Speicherstelle $00AABB (HHMMLL) der AP33 poken. Hierzu poken wir zunächst $00 in das High-Byte, $AA in das Middle-Byte und $BB in das Low-Byte des 20-Bit-Adreßregister. Nunmehr poken wir $CC in das 8-Bit-Datenregister, womit $CC in $00AABB übertragen wird.

Peek-Beispiel: Nehmen wir umgekehrt an, wir wollen später wieder das Byte $CC aus der Speicherstelle $00AABB der AP33 peeken. Hierzu poken wir zunächst wieder die Adresse in das 20-Bit-Adreßregister und peeken dann den Wert $CC aus dem 8-Bit-Datenregister.

Paritätsprüfung: In Wirklichkeit ist es etwas komplizierter. Erstens muß man zwischen der alten 64K-Chip- und der neuen 256K-Chip-Karte unterscheiden. Zweitens sollte man von der bei der AP33 implementierten Möglichkeit Gebrauch machen, die Datenbytes auf Parität zu überprüfen. Es wird das Verfahren der „Even parity" (= geraden Parität) verwendet. *Gerade Parität* besagt bei einem 8-Bit-Byte, z.B. %10110110, folgendes: Man zählt die Einserbits zusammen, hier 5 Einserbits, und prüft, ob sich eine gerade oder eine ungerade Zahl ergibt. Da 5 eine ungerade Zahl ist, wird das Paritätsbit auf 1 gesetzt. Bei %11110000 würde sich die Summe 4 ergeben, also eine gerade Zahl, so daß hier das Paritätsbit auf 0 gesetzt wird. Somit folgt, daß bei gerader Parität die Summe der Einserbits des 8-Bit-Bytes *und* des Paritätsbits zusammen immer eine gerade Zahl ergeben. Wenn man also bei der AP 20 unter Verwendung der Paritätsprüfung beispielsweise das Byte %11000010 von der Karte liest und das Paritätsbit ist *nicht* auf 1 gesetzt, so liegt ein Übertragungsfehler vor.

2. AP33 und ProDOS

Die AP33 ist wahrscheinlich die erste große RAM-Karte, für die ein RAM-Disk-Driver für ProDOS entwickelt wurde. Daneben liegen die bereits bekannten RAM-Disk-Driver für DOS 3.3, CP/M und Pascal 1.1 vor, auf die wir aus Platzgründen hier nicht näher eingehen werden. Der ProDOS-RAM-Disk-Driver wurde im April dieses Jahres von Volker Baumgarte entwikkelt und mir freundlicherweise von der Firma IBS als Quellcode zur Prüfung zugesandt. Leider springen einem bei Fremdprogrammen die Fehler leichter ins Auge als bei eigenen Programmen. Tatsächlich entdeckte ich prompt einen „Schußligkeitsfehler", der zur Folge hatte, daß die Blocks 3 und 4 des Volume-Directory nicht mit den korrekten Vorwärts- und Rückwärtszeigern versehen wurden. Dies ist inzwischen behoben. Sollte ein Leser einen der ersten Prototypen der Karte zufällig noch mit dem alten ProDOS-Driver erhalten haben, so möge er bei der AP33 bitte folgenden Test durchführen:

```
10 FOR X = 1 TO 51:
20 PRINT CHR$(4) "SAVE XXX"X: NEXT
```

Sollte hierbei ein I/O-Fehler auftreten, so fordern Sie bitte die neue Driver-Diskette an.

Der IBS-ProDOS-Driver (namens INSTAL oder MEGAWARP) läuft unter allen ProDOS-Versionen, was keineswegs eine Selbstverständlichkeit ist. Der berüchtigte Test für die Überprüfung der Blockobergrenze (maximal 280 Blocks) braucht übrigens nicht gepatcht zu werden, weil sich dieser Test am Anfang des *Disk-II*-Drivers befindet, der bei einem *RAM-Disk*-Driver gar nicht zum Tragen kommt. Um den MEGAWARP-Driver zu installieren, bootet man zunächst eine beliebige ProDOS-Version von der ProDOS-Systemdiskette und startet dann mit BRUN INSTAL den Driver von der IBS-Diskette. Das MEGAWARP-Programm

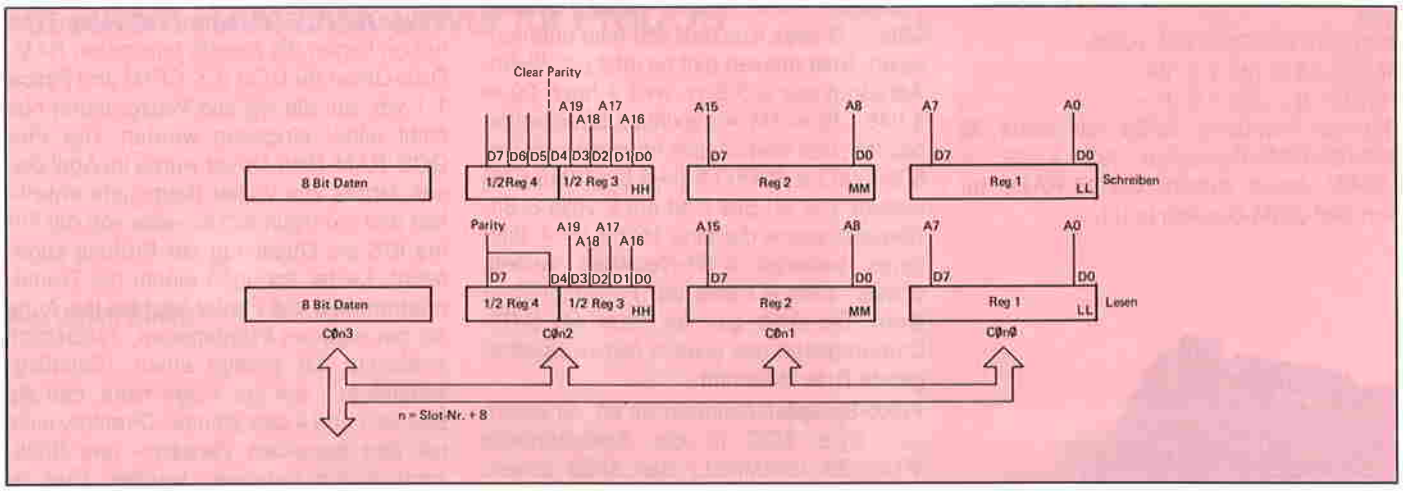
(1) fragt nach der Nummer n des Slots, in dem sich die AP33 befindet,
(2) ermittelt die Bestückung und ob 64-KBit- oder 256-KBit-Chips vorliegen,
(3) löscht die gesamte RAM-Karte (LDA #0 STA $000000 bis $0HMMLL),
(4) installiert den Driver ab $Cs00 und $C800 und
(5) legt das Volume-Directory an.

3. AP33 und Accelerator IIe

Wenn die AP33 beispielsweise im Slot 4 steckt – jeder Slot außer Slot 0 wäre zulässig – so beginnt der ProDOS-RAM-Disk-Driver bei $C400 mit einem JMP nach $C800, denn neben dem eigentlichen RAM der AP33 verfügt man bei dieser Karte über 256 Bytes Slot-RAM ($Cs00-$CsFF) sowie über I/O-RAM im Bereich $C800-$CFFE, der mit BIT $CFFF lese/schreibfähig gemacht wird.

Die Accelerator IIe, die im Peeker, Heft 1/1984, besprochen wurde, taktet jedoch von 3,5 auf 1 MHz herunter, wenn Interface-Adressen ($C000-$CFFF) angesprochen werden. Deshalb sollte man bei Verwendung der Accelerator IIe in Verbindung mit der AP33 den Driver in einem Nicht-Interface-Bereich ansiedeln.

Unser Programm „MEGAWARP.NEU" in den Versionen MEGAWARP.REL (relokativ) und MEGAWARP.9900 (nicht relokativ) dient diesem Zweck und führt zu einer *Verdopplung* der Datenübertragungsrate („2 MHz") gegenüber dem ursprünglichen IBS-Driver („MEGAWARP.ALT"). Die „2 MHz" erklären sich dadurch, daß etwa die Hälfte der Driver-Befehle im schnellen Accelerator-RAM mit ca. 3,32 MHz ausgeführt werden, während beim eigentlichen AP33-Zugriff auf 1 MHz heruntergetaktet wird. Man beachte, daß unsere Driver nur dann sinnvoll sind, wenn man neben der AP33 die Accelerator IIe besitzt, denn ohne letztere Karte hat MEGAWARP.NEU keine Geschwindigkeitsvorteile. Ferner sei darauf hingewiesen, daß unsere Driver nur für die 256-KBit-ICs gedacht sind und daß auf den Initialisierungsteil verzichtet wurde, der bereits in INSTAL (= „MEGAWARP.ALT") enthalten ist. Man starte also erst MEGAWARP.ALT (mit BRUN INSTAL) und dann MEGAWARP.NEU (mit BRUN MEGAWARP.REL oder BRUN MEGAWARP.9900). Beide Module befinden sich auf der Peeker-Sammeldisk, doch ist aus Platzgründen nur MEGAWARP.9900 gelistet. Wenn die AP33 nicht im Slot 4 steckt, muß man den Quellcode entsprechend abändern (Labels SLOT4 und SLOTC4)



4. Geschwindigkeitstests

Im einzelnen wurden folgende Geschwindigkeitstests durchgeführt:

a) MEGAWARP.ALT mit/ohne Accelerator

b) MEGAWARP.NEU mit/ohne Accelerator

c) Apple-IIe-64K-Karte mit ProDOS-RAM-Disk-Driver mit/ohne Accelerator

Bei der 64K-Karte hängt im Gegensatz zur AP33 die Datenübertragungsrate davon ab, welche Bereiche der 64K-Karte beim RAM-Disk-Zugriff tangiert werden.

Näheres über die Testergebnisse kann man den Listings entnehmen.

„High-Level"-Test

Dieser Test sollte den BASIC.SYSTEM-RAM-Disk-Zugriff mit BSAVE und BLOAD messen. Wenn man von den Seek-Befehlen sowie den Directory- und Index-Block-Zugriffen absieht, wurde als Grobwert 1M übertragen. MEGAWARP.NEU ist hier genauso schnell wie MEGAWARP.ALT, wenn keine Accelerator eingesetzt wird. Unter Verwendung der Accelerator ist die Datenübertragungsrate jedoch beachtlich (20.7K/s bei MEGAWARP.ALT ohne ACCEL gegenüber 37.9K/s bei MEGAWARP.NEU mit ACCEL).

Wie ich dem alten IBS-MEGAWARP-Driver entnehmen konnte, wurde der Seek- bzw. Status-Befehl durch „LDA COMMAND LSR BCC WRITE" wie ein Write-Befehl behandelt. Abgesehen davon, daß dies „ins Auge gehen" kann, wird unnötige Zeit vertrödelt. Der nachfolgende Spezialtest mag dies verdeutlichen:

```
10 N = 100
20 PRINT CHR$(4)
   "RENAME/MEGAWARP,/M"
30 PRINT CHR$(4) "CREATE/M/D,TDIR"
40 PRINT CHR$(4) "PREFIX/M/D"
50 FOR X = 1 TO N:
60 PRINT CHR$(4) "SAVE X"X: NEXT
70 FOR X = 1 TO N
80 PRINT CHR$(4) "DELETE X"X: NEXT
```

Ohne Accelerator benötigt MEGAWARP.ALT hierfür 67,5s im Gegensatz zu MEGAWARP.NEU mit nur 65s. Die Differenz von 2,5s ist auf den Seek- bzw. Status-Befehl zurückzuführen.

„Low-Level"-Test

Dieser Test sollte den MLI-RAM-Disk-Zugriff mit den Blockread- und Blockwrite-

Ablage Bearbeiten Große Verlag Zeichensatz Stil Hilfsmittel

pandabooks

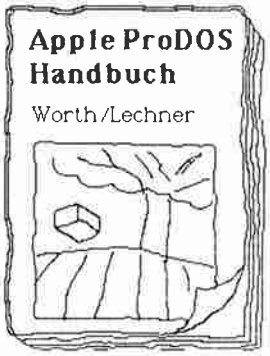
Bismarckstr. 67, D-1000 Berlin 12 (030) 342 88 00

Gratis!
AppleQuick
Das Apple II-Brevier für häufig gebrauchte Adressen und Befehle! 64 Seiten über Applesoft, DOS, Pascal, CP/M. Gleich anfordern!

Pandabooks!

Die Qualität kommerzieller Arcadespiele läßt sich mit APPLESOFT BASIC alleine nicht erreichen. Jeffrey Stanton führt in die Eigenarten der hochauflösenden Apple-Grafik ein und präsentiert schließlich eine Reihe extrem schneller Assembler-Routinen, mit denen Sie viele Effekte bekannter Spiele selbst programmieren können. Gute BASIC-Kenntnisse werden vorausgesetzt, eine Einführung in Assembler-Programmierung wird gegeben.

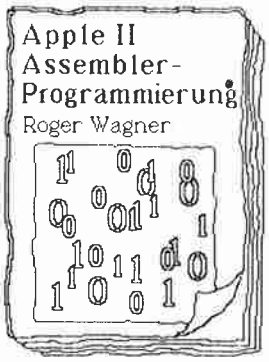
3-89058-006-8 299 Seiten DM 49,--
komplett mit Disk DM 89,--



Don Worth und Peter Lechner sind die Autoren von "Beneath Apple DOS", der Bibel aller DOS 3.3-Benutzer. In ihrem neuen Buch haben sie sich ProDOS vorgenommen und mit der ihnen eigenen Gründlichkeit zerlegt. Ausführliche Beschreibung der Arbeitsweise, der Dateitypen, des "MLI" und des "BASIC SYSTEM". Das Standardwerk für jeden ProDOS-Anwender.

3-89058-036-X ca. 250 S. DM 46,--

Das Assembler-Lehrbuch für BASIC-Kenner.
Roger Wagner, der Autor vieler bekannter Software-Pakete, schrieb eine monatliche Kolumne über Assembler-Programmierung in der Apple-Zeitschrift SOFTALK.
Der vorliegende Band faßt diese Reihe, korrigiert und erweitert, zusammen: Eine stufenweise Einführung in die Befehle und Strukturen der 6502-Assemblersprache, mit vielen Beispielen von der einfachen Tongenerierung bis zum Diskettenzugriff.
3-89058-003-3 277 Seiten DM 48,--
komplett mit Disk DM 89,--



Ein Simulationsprogramm, das Sie in das Innere des 6502 Mikroprozessors führt. Sie sehen auf dem Bildschirm, wie die einzelnen Instruktionen in Zeitlupe ausgeführt werden, wie sich die Register und die Flags verändern. Ein unverzichtbares Hilfsmittel beim Erlernen der Assembler-Programmierung, danach ein wertvolles Werkzeug beim Testen Ihrer eigenen Programme. Komplett mit einem 6502 Editor/-Assembler, deutschem Handbuch (150 Seiten) und über 30 Beispielprogrammen.

3-89058-019-X DM 129,--

Die beiden Autoren haben zusammen mehr als 50 Jahre Amateurfunk-Erfahrung und präsentieren hier mehr als 20 BASIC-Programme, die jeden Funkamateur interessieren: QTH-KENNER, DÄMMERUNGSLINIE ,DX-RECHNER ,DOPPEL-FAHNDER, NACHWEISSCHREIBER, ALLZWECK-CONTEST-LOGGER, FIELD-DAY-LOGGER, WETTBEWERB-LOGGER u.v.a.m. Das Buch enthält die kompletten Listings für Apple II und C64.

3-89058-027-0 ca. 256 S. DM 48,--
komplett mit Disk DM 89,--

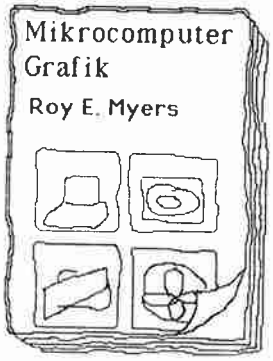
Eine detaillierte Beschreibung der Apple II und Apple IIplus Schaltungen. Wenn Sie Ihren Apple selbst reparieren, Interface-Karten oder Schaltungserweiterungen entwerfen oder einfach nur besser über das Innenleben Ihres Apple Bescheid wissen wollen - dieses Buch bietet Ihnen eine Fülle an Informationen, Schaltpläne und Zeitdiagramme, Theorie und praktische Tips.

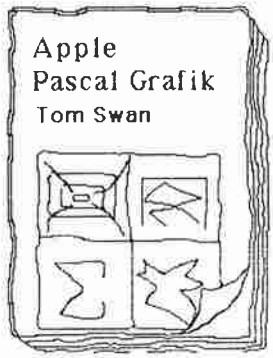
3-89058-012-2 215 S. A4 DM64,--

Endlich anspruchsvolle Apple-Grafik für BASIC-Programmierer. Mikrocomputer Grafik

- enthält fast 80 lauffertige BASIC-Programme, die die beschriebenen Konzepte illustrieren.
- beschreibt "Hidden Line"- und "Hidden Surface"-Techniken, Skalierung, Rotation und Translation von Grafiken
- bietet eine Einführung in die Animationstechnik

3-89058-000-9 292 Seiten DM 49,--
komplett mit Disk DM 89,--





22 Pascal-Programme, mit denen Sie die Grafik-Möglichkeiten Ihres Apple voll ausschöpfen:
"Designer" ermöglicht es Ihnen, eigene Zeichensätze zu entwerfen.
"Credit" unterstützt Sie beim Entwerfen komplexer Bildschirm-Grafiken
"Printfoto" bringt Ihre Entwürfe aufs Papier.
Darüberhinaus bietet das Buch eine Fülle fertiger Prozeduren, die Sie zeitsparend in Ihre eigenen Programme einbauen können

3-89058-009-2 280 Seiten DM 49,--
komplett mit Diskette DM 89,--

In allen Buchhandlungen und Computershops oder direkt von Pandabooks, Bismarckstr. 67, 1000 Berlin 12, (030) 342 88 00

Bestellcoupon

Name:
Anschrift:

O V-Scheck liegt bei (spesenfreie Lieferung)
O per Nachnahme (zzgl. Versandspesen)

Menge	Titel	Preis
1	AppleQuick	gratis

Österreich: OD Landesverlag, Linz * Schweiz: Thali AG, Hitzkirch

Routinen messen. Es wurde wie beim „High-Level"-Test 1M übertragen (s. SPEEDTEST). Während *ohne* Accelerator der Geschwindigkeitsunterschied zwischen MEGAWARP.ALT und MEGAWARP.NEU praktisch gleich null ist, steigt *mit* der Accelerator-Karte die Übertragungsrate überproportional an. Beim Blockread wurden „nur" 81.9K/s übertragen, während beim Blockwrite satte 128K/s erzielt werden konnten. Damit dürfte die AP33 in Verbindung mit der Accelerator IIe schätzungsweise zehnmal schneller als die Profile-Festplatte sein, bei der eine Accelerator praktisch wirkungslos ist. (Es wäre nett, wenn ein Peeker-Leser, der eine Profile besitzt, uns seine Meßergebnisse anhand des SPEEDTEST mitteilte.)

5. Fazit

Die AP33 ist eine RAM-Karte der neuen Generation. Mit 1024K hat man die doppelte Kapazität des „Fat Mac" (= 512K-Macintosh), wobei die Mac-512K-Speichererweiterung (Listenpreis!) etwa soviel wie eine 1024K-AP33 kostet. Nimmt man noch die Accelerator IIe hinzu, so verfügt man über traumhafte Übertragungsraten. Sollte es zufällig einen 68000-Programmierer geben, der einen 512K-Mac besitzt, so wird er gebeten, uns die Ergebnisse des entsprechenden Tests mitzuteilen.

MEGAWARP-Geschwindigkeitstest

```
MEGAWARP.ALT = Driver der Firma IBS, Bielefeld
MEGAWARP.NEU = unsere Version für Accelerator
               a) MEGAWARP.REL  (relokativ - nicht selbstmodifizierend)
               b) MEGAWARP.9ØØØ (nicht relokativ - selbstmodifizierend)
```

1. "High-Level"-Test

1.1. BSAVE-BLOAD-Testprogramm

16 * 32768 * 2 für BSAVE/BLOAD = 1.Ø48.576 = 1 Megabyte.
Grobwert ohne Berücksichtigung der Directory- und Index-Blockzugriffe.

```
1Ø FOR X = 1 TØ 16: PRINT CHR$(4) "BSAVE XXX,A$1ØØØ,L$8ØØØ": NEXT
2Ø FOR X = 1 TO 16: PRINT CHR$(4) "BLOAD XXX": NEXT
3Ø PRINT CHR$(7)
```

1.2. Testwerte in Kilobytes/Sekunde

	MEGAWARP.ALT	MEGAWARP.NEU	64K-Karte (1)	64K-Karte (2)
Ohne ACCEL	2Ø.7	2Ø.7	26.2	21.5
Mit ACCEL	25.9	37.9	49.9	35.3

2. "Low-Level"-Test

2.1. BLOCK-READ/WRITE-Testprogramm

```
BLOAD SPEEDTEST
CALL -151
3Ø3: 8Ø (für Read-Test)
3ØØG
3Ø3: 81 (für Write-Test; zerstört RAM-Disk-Inhalt)
3ØØG
Der Speedtest kann auch für Festplatten und Disk-II-Drives
verwendet werden. Zuvor Parameter (Unit usw.) entsprechend ändern.

                  1              ORG  $Ø3ØØ
                  2     *
                  3     * SPEEDTEST
                  4     * ========
                  5     *
                  6     * 32 mal 64 Blocks lesen
                  7     * oder schreiben = 2Ø48 Blocks
                  8     * = 1M = 1.Ø48.576 Bytes = 2↑2Ø
                  9     *
Ø3ØØ: 4C ØA Ø3   1Ø              JMP  START
Ø3Ø3: 8Ø         11    RDWR      HEX  8Ø          ;Read (81=Write)
Ø3Ø4: 4Ø         12    UNIT      HEX  4Ø          ;S4,D1 (6Ø=S6,D1)
Ø3Ø5: 2Ø         13    MAL       DFB  32          ;Wieviel mal?
Ø3Ø6: 65 ØØ      14    ERSTER    DA   1Ø1         ;Erster Block
Ø3Ø8: A4 ØØ      15    LETZTER   DA   164         ;Letzter Block
Ø3ØA: AD Ø3 Ø3   16    START     LDA  RDWR
Ø3ØD: 8D 2B Ø3   17              STA  COMMAND
Ø31Ø: AD Ø4 Ø3   18              LDA  UNIT
Ø313: 8D 5Ø Ø3   19              STA  UNITNO
Ø316: AD Ø5 Ø3   2Ø              LDA  MAL
Ø319: 8D 55 Ø3   21              STA  MALCNT
Ø31C: AD Ø6 Ø3   22    LOOP1     LDA  ERSTER
Ø31F: 8D 53 Ø3   23              STA  BLOCKL
Ø322: AD Ø7 Ø3   24              LDA  ERSTER+1
Ø325: 8D 54 Ø3   25              STA  BLOCKH
Ø328: 2Ø ØØ BF   26    LOOP2     JSR  $BFØØ       ;MLI
Ø32B: 8Ø         27    COMMAND   HEX  8Ø
Ø32C: 4F Ø3      28              DA   COUNT
Ø32E: BØ 26      29              BCS  ERROR
Ø33Ø: EE 53 Ø3   3Ø              INC  BLOCKL
Ø333: DØ Ø3      31              BNE  SUBTRACT
Ø335: EE 54 Ø3   32              INC  BLOCKH
Ø338: 38         33    SUBTRACT  SEC
Ø339: AD Ø8 Ø3   34              LDA  LETZTER
Ø33C: ED 53 Ø3   35              SBC  BLOCKL
Ø33F: AD Ø9 Ø3   36              LDA  LETZTER+1
Ø342: ED 54 Ø3   37              SBC  BLOCKH
Ø345: BØ E1      38              BCS  LOOP2
Ø347: CE 55 Ø3   39              DEC  MALCNT
Ø34A: DØ DØ      4Ø              BNE  LOOP1
Ø34C: 4C 3A FF   41              JMP  $FF3A       ;BELL
Ø34F: Ø3         42    COUNT     HEX  Ø3
Ø35Ø: 4Ø         43    UNITNO    HEX  4Ø          ;S4,D1
Ø351: ØØ 4Ø      44    PUFFER    DA   $4ØØØ       ;-$41FF
Ø353: ØØ         45    BLOCKL    HEX  ØØ
Ø354: ØØ         46    BLOCKH    HEX  ØØ
                 47    *
Ø355: ØØ         48    MALCNT    HEX  ØØ
Ø356: 4C DA FD   49    ERROR     JMP  $FDDA       ;PRBYTE

89 Bytes
```

RAM-Disk-Driver für ProDOS

Allgemeine Richtlinien

Nachfolgend soll unabhängig von der AP33 und der 64K-Karte, für die bereits RAM-Disk-Driver vorliegen, kurz charakterisiert werden, was im einzelnen beachtet werden muß, wenn man einen RAM-Disk-Driver für eine neue RAM-Karte implementieren will.

Welche Speicherkapazität?
Eine RAM-Karte für ProDOS kann theoretisch eine Speicherkapazität von 32M haben. 64K dürfte jedoch die praktische Untergrenze und 1M die praktische Obergrenze sein.

Welcher Slot und Drive?
Sinnvollerweise wird man die RAM-Disk mit demselben Slot s ansprechen (CATALOG, Ss), in dem die RAM-Karte steckt, obwohl dies nicht unbedingt erforderlich ist. In der PRODOS Global Page befinden

Wir vertreten unter anderem folgende Firmen in Deutschland:

Mountain™ Computer Inc.

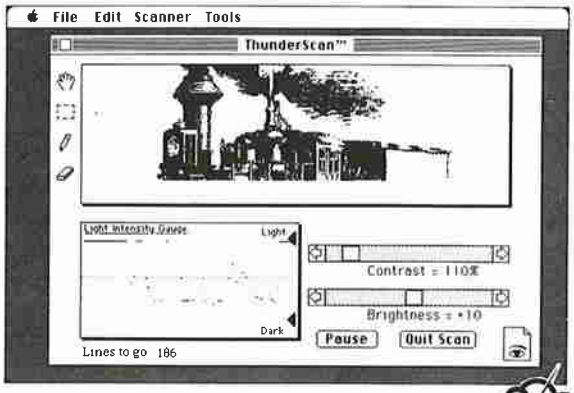
Händleranfragen erwünscht.

pandasoft Dr.-Ing. Eden

Uhlandstraße 195 · 1000 Berlin 12 · Mo–Fr 10–18 Uhr, Sa 10–13 Uhr
Telefon: 0 30/31 04 23 · Telex: 1 85 859

ThunderScan.™

Ein neues optisches Lesegerät, das beliebige Vorlagen in MacPaint überträgt: Fotos, Zeichnungen, Landkarten und Illustrationen werden in den Apple-Imagewriter eingespannt und von einem Lesekopf, der das Farbband ersetzt, abgetastet.



- 32 Graustufen
- 80 Punkte/cm Auflösung
- Übertragungsmaßstab 25 % – 400 %
- Vorlagen bis 20 x 25 cm
- Nachträgliche Veränderung des Kontrasts und der Helligkeit.

ThunderScan

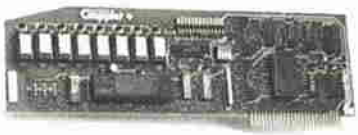
pandasoft Dr.-Ing. Eden

Uhlandstraße 195 · 1000 Berlin 12 · Mo–Fr 10–18 Uhr, Sa 10–13 Uhr
Telefon: 0 30/31 04 23 · Telex: 1 85 859

Druckerinterfaces für Apple II+/e/c/III Interfaces auf dem **neuesten Stand der Technik. Kompatibel** mit allen gängigen Druckern wie: APPLE, EPSON, STAR, NEC, OKIDATA usw. Passende Treibersoftware wird über Dip-Switch ausgewählt.

Grafikfähiges Druckerinterface das keine Wünsche mehr offen läßt. Über **2 Dutzend Kommandos** ermöglichen die volle Kontrolle über alle Möglichkeiten Ihres Druckers. Jetzt auch mit **IIe Features: Double Hires Grafics** und **80 Zeichen Dump** mittels Druckerpuffer nachrüstbar über Bufferboard.

Besitzt alle Vorzüge des Grappler +, hat aber zusätzlich einen integrierten **16 K Druckpuffer,** der auf **32 oder 64 K aufrüstbar** ist.



Serielles Druckerinterface speziell für den **Apple Imagewriter.**

Seriell-nach-Parallel-Wandler für den IIc im Kabel integriert.

wie Hotlink, jedoch zusätzlich Imagewriter Emulation und Grafik Software-Diskette.

pandasoft Dr.-Ing. Eden

Uhlandstraße 195 · 1000 Berlin 12 · Mo–Fr 10–18 Uhr, Sa 10–13 Uhr
Telefon: 0 30/31 04 23 · Telex: 1 85 859

Sie haben einen Apple...

wir haben die Software...

und die Hardware...

wir haben die Bücher...

und die Zeitschriften*...

***Fordern Sie unseren Gratiskatalog an!**

ALLES FÜR DEN APPLE II+, IIe, IIc UND MACINTOSH

pandasoft Dr.-Ing. Eden

UHLANDSTR. 195 · D-1000 BERLIN 12
TEL.:(030) 310 423 · TELEX: 18 58 59

Automatisierter Apple Fachhandler MICROSOFT Distributor

Ich besitze einen Apple. Bitte schicken Sie mir Ihren kostenlosen Katalog.
Name: .....
Adresse: .....

sich ab $BF10-$BF2F die Einsprungadressen für die diversen Volume-Driver. Das Installierungsprogramm muß dort den Vektor für den entsprechenden Slot und Drive eintragen, die Device-Anzahl ($BF31) erhöhen und die Device-Liste ab $BF32 um das Device-Bit-Muster für die RAM-Disk erweitern (s. „ProDOS für Aufsteiger", Bd. 1, S. 51ff.). Eine RAM-Disk muß mit *einem und nur einem* Drive angesprochen werden. Befände sich beispielsweise die RAM-Karte in Slot 4, so könnte man zwar bei $BF18 für Drive 1 *und* bei $BF28 für Drive 2 quasi einen Doppeleintrag vornehmen, doch würde dann das MLI u.U. „durchdrehen", weil *derselbe* Volume-Name für 2 Units gelten würde.

Was bedeutet Formatierung?

Das Installierungsprogramm sollte die Formatierung als gesonderte Option zulassen, so daß man den Driver nach versehentlichem Neubooten wieder anschließen kann, ohne daß der Inhalt der RAM-Disk zerstört wird. Der 64K-RAM-Disk-Driver der Firma Apple verfügt leider nicht über diese Option. Die eigentliche „Formatierung" sollte zunächst alle Blocks (= 512-Byte-Bereiche der RAM-Karte) oder zumindest die Blocks 2-6 auf null setzen und dann die Blocks 2-6 entsprechend der Kapazität der RAM-Karte initialisieren. Blocks 2-5 enthalten das Volume-Directory. Das Volume-Directory muß 4 Blocks umfassen, weil sonst das MLI oder das BASIC.SYSTEM „durchdrehen" kann. Auch dies ist von der Firma Apple bei ihrem 64K-Karte-Driver nicht beachtet worden, so daß sich einige Anwenderprogramme wie Mailmerger (s. Peeker, Heft 4/1985, S. 73) mit I/O-Error oder End-of-Data „verabschieden". Ferner muß der Volume-Bit-Map-Block 6 – bei RAM-Karten üblicherweise nur ein einziger Block – angelegt werden. (Ein Beispiel für die Anlage der Blocks 2-5 findet sich in „ProDOS für Aufsteiger", Bd. 2, Kap. 3.5.)

Wohin mit dem Driver?

Die Firma Apple hat für sich die gesamte 64K-Karte „gepachtet" und einen Teil des eigenen 64K-Karte-Drivers dort untergebracht, der je nach ProDOS-Version einmal hier und einmal dort liegt. Fremdentwickler haben es hier schwerer, da der Driver nur entweder in dem Bereich unterhalb vom BASIC.SYSTEM, d.h. unterhalb von $9A00 (vgl. MEGAWARP.NEU), *und/oder* im Interface-Bereich ($Cs00-$CsFF sowie $C800-$C8FE, vgl. MEGAWARP.ALT) *und/oder* auf der RAM-Karte selbst liegen kann (vgl. 64K-Karte-Driver). Wenn

2.2. Testwerte in Kilobytes/Sekunde

	MEGAWARP.ALT		MEGAWARP.NEU	
	Read	Write	Read	Write
Ohne ACCEL	38.2	53.6	38.3	53.6
Mit ACCEL	40.1	56.6	81.9	128.0

3. MEGAWARP.9900 und MEGAWARP.REL für Accelerator-Besitzer

MEGAWARP.9900 und MEGAWARP.REL sind funktionsgleich. MEGAWARP.9900 ist jedoch nicht relokativ und etwas schneller ("getunt").
1. AP33 (256-KBit-Chips) in Slot 4 installieren
2. ProDOS (beliebige Version) booten
3. BRUN INSTAL (von mitgelieferter IBS-Diskette)
4. BRUN MEGAWARP.9900 oder BRUN MEGAWARP.REL
Nunmehr hat man eine erheblich gesteigerte Übertragungsrate.
Für andere Slots (als Slot 4) Quellcode entsprechend ändern.
MEGAWARP.REL ist aus Platzgründen nicht gelistet.

```
                 1              ORG  $1FBA
                 2    *
                 3    * MEGAWARP.9900
                 4    * ==============
                 5    *
                 6    * "Speed-Version" von MEGAWARP.NEU
                 7    * Nicht relokativ wegen der Label
                 8    * WRPAGE1 und RDPAGE2, deren
                 9    * Adressen gepokt werden.
                10    * Slot-Adresse (hier Slot 4)
                11    * modifizierbar.
                12    *
                13    SLOT4    EQU  $40        ;4*16
                14    SLOTC4   EQU  $C400
                15    *
                16    * Move-Routine
                17    * =============
                18    *
                19    GETBUFR  EQU  $BEF5
                20    SLDR     EQU  $BF10
                21    COUT     EQU  $FDED
                22    *
                23    * 1 Page über Getbuffer-Routine anfordern
                24    *
1FBA: A9 01     25             LDA  #1
1FBC: 20 F5 BE  26             JSR  GETBUFR
1FBF: B0 24     27             BCS  ERROR
1FC1: C9 99     28             CMP  #$99
1FC3: D0 20     29             BNE  ERROR
                30    *
                31    * Driver nach $9900 oder tiefere Page moven
                32    *
1FC5: A0 00     33             LDY  #0
1FC7: B9 00 20  34    MOVE     LDA  $2000,Y    ;fest!
1FCA: 99 00 99  35             STA  $9900,Y    ;fest!
1FCD: C8        36             INY
1FCE: D0 F7     37             BNE  MOVE
                38    *
                39    * Slot-Drive-Tabelle nach $C400 absuchen
                40    *
1FD0: B9 10 BF  41    SLDR1    LDA  SLDR,Y
1FD3: AA        42             TAX             ;X=LL
1FD4: C8        43             INY
1FD5: B9 10 BF  44             LDA  SLDR,Y     ;A=HH
1FD8: E0 00     45             CPX  #$00       ;LL=00?
1FDA: D0 04     46             BNE  SLDR2
1FDC: C9 C4     47             CMP  #>SLOTC4
1FDE: F0 0A     48             BEQ  SLDR3
1FE0: C8        49    SLDR2    INY
1FE1: C0 10     50             CPY  #$10
1FE3: D0 EB     51             BNE  SLDR1
1FE5: A9 87     52    ERROR    LDA  #$87       ;Bell
1FE7: 4C ED FD  53             JMP  COUT
                54    *
                55    * GETBUFR-HIMEM in SLDR eintragen
                56    *
1FEA: A9 99     57    SLDR3    LDA  #$99       ;$9900
1FEC: 99 10 BF  58             STA  SLDR,Y
1FEF: 88        59             DEY
1FF0: A9 00     60             LDA  #$00       ;00
1FF2: 99 10 BF  61             STA  SLDR,Y
                62    *
                63    * In System-Bit-Map die Page $99
                64    * als belegt markieren
                65    *
1FF5: A9 7F     66             LDA  #%01111111
1FF7: 8D 6B BF  67             STA  $BF6B      ;$9800-$9FFF
                68    *
```

ccp datentechnik

Neu Neu Neu Neu Neu

640 KByte-Drives für den Apple //c!!

- 5¼- od. 3½-Zoll-Format (Teac FD55/35-F)
- FD55-F umschaltbar auf 35/40 Track
- Anschluß an die externe Laufwerkbuchse
- Durch Einbauplatine (kein Löten) 640 KByte im Direktzugriff
- Einfache Anpassung für DOS 3.3, UCSD-Pascal und PRODOS durch menügeführten Patch
- Anpassung von CP/M in Verbindung mit einer Z 80-Zusatzplatine in Vorbereitung
- anschlußfertig im Gehäuse . . . . . **DM 1090,–**

Festplatten für Apple II (//e)

- 5¼ Zoll-Format (Slimline)
- Booten direkt von der Festplatte in DOS 3.3, UCSD-Pascal, PRODOS und CP/M 2.2 / 3.0
- Gemischtbetr. mit 35/40/80/160 Track-Drives
- Copy- und Install-Programme im Lieferumfang
- Umfangreiches Manual
- z. B. 12 MB form. incl. Netzteil u. Contr., anschlußfertig an Ihren Apple . . . . . **DM 3835,–**

640 KByte-Drives für Apple II (//e)

- 5¼- od. 3½-Zoll-Format (Teac FD55/35-F)
- FD55-F umschaltbar auf 40 Track (Apple kompatigel)
- Installationssoftware für DOS 3.3, UCSD-Pascal, CP/M 2.2, CP/M 2.23 (60K), PRODOS, AP22, ALS CP/M+
- Umfangreiches Handbuch
- Anschlußfertige Auslieferung incl. Contr. und 2 Drives
- Diskstation 55II (2 Teac FD55-F, 1.2 MB) . . . **DM 1498,–**
- Diskstation 35II (2 Teac FD35-F, 1.2 MB) . . . **DM 1580,–**

Alles für Ihren Apple

Info bei:

ccp-datentechnik

Herderstraße 12 – 2000 Hamburg 76
Telefon 040/225676

FORTH-SYSTEME

FORTH ist eine schnelle Programmiersprache, sie ermöglicht Programmtransfer zwischen APPLE und anderen Rechnern.

Unser Buchangebot zu FORTH:

Programmieren in FORTH v. Leo Brodie dt.	48,— DM
Thinking FORTH v. L. Brodie engl.	65,— DM
Mastering FORTH, Einführung in F83	78,— DM
FORTH **Encyclopedia**	98,— DM
Expertensystem in FORTH	98,— DM

FORTH-Programme für **APPLE II/e/c/+:**

fysFORTH 0.3 mit 600seitigem Handbuch	198,— DM
MasterFORTH83 v. Micro Motion Grundsys.	398,— DM
Floatingpoint, Hires, Modules	350,— DM
LMI FORTH unter **PRODOS** für IIe/c mit Banksupport, Editor, Tools in Kürze 1b.	398,— DM

FORTH-Programme für **APPLE Macintosh:**

NEON! v. Kriya Systems, **Objektorientierte Programmierung,** Einführungspreis	999,— DM
Mac FORTH v. Creative Solutions ab	680,60 DM
MasterFORTH83 v. Micro Motion Grundsys.	498,— DM
Floatingpoint und Modulesextension	398,— DM

FORTH-SYSTEME Angelika Flesch
Pf. 1226, 7820 Titisee-Neustadt, Tel. 07651/1665

MICROMINT

MICROMINT VOLLTREFFER

LASAR 16
IBM 256 K, 2 × TEAC B FDD, Contr. color Graphik, Multifunktionscard, Tastatur, Monitor Netzteil 15 A **4.678;**

LASAR ZE
– Apple comp. 64 K + 12 K ROM + 6502 + Z 80 A 80 Z sw Tastatur **1.290;**

Außerdem volles Rückgaberecht innerhalb 14 Tagen ohne Begründung.

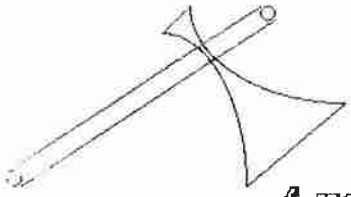
	Apple	IBM
● Mehrzweckklappgehäuse lt. Abb.	147,–	147,–
● Schaltnetzteile Apple 5 A/IBM 15 A	115,–	238,–
● Profitastatur dtsch. LASAR 2000	291,–	291,–
● Interface ab	75,–	148,–
● Monitor 22 Mhz incl. Fuß, bernstein	289,–	289,–

Kaufgarantie/Tiefstpreisgarantie/1A Qualität: 100 % kompatibel inkl. Systemsoftware Made by Micromint: Apple II 495,–, Apple IIe 695,–, IBM 895,– Fertigplatinen. Tragbare Gehäuse für 7-Zoll/9-Zoll-Monitore 595,– DM inkl. Tastaturen. Winchester 27 MB auf Anfrage 1A First Class Controller bis 140 MB 935,– DM.

Generalimporteur MICROMINT Computer GmbH
Hochdahler Straße 151, 4006 Erkrath 2
Telex 8589305 mcm

☎ **02104/33024**

WIR MACHEN KEINEN HECKMECK



(HACMAC)

Weil wir Ihren Mac professionell von 128KB auf 512 KB umrüsten.
Schicken oder bringen Sie uns Ihren Computer. Wir garantieren die Rücksendung innerhalb 24 Stunden.

Nach der Umrüstung haben Sie bei uns 1/2 Jahr Garantie!

Wir vermitteln Ihnen auch gerne einen Händler in Ihrem Raum, der Ihnen unsere Umrüstung einbaut.

Der Preis : DM 1467.00 incl. MWST.

CP/D oHG, Flügelstr. 47
4000 Düsseldorf, Tel:0211-776270

der Driver im Slot-Bereich $Cs00-$CsFF liegt, muß dafür Sorge getragen werden, daß nach PR#s nicht der Driver aktiviert wird (vgl. MEGAWARP.ALT). Zumindest sollte man bei $Cs00 einen Sprung zu einer Fehlerroutine implementieren, etwa
BIT $C082
JMP $FF59 (Reset)
(vgl. hierzu Peeker, Heft 5/1985, S. 14, „SLOTRAMDISK").

Welche Parameter auswerten?

Vor dem Sprung zu dem RAM-Disk-Driver sind vom MLI in der Zeropage folgende Werte abgelegt worden:
$0042: Befehl (0=Seek, 1=Read, 2=Write, 3=Format)
$0043: Unit-Nummer (z.B. 40 für S4, D1)
$0044: LLHH des 512-Byte-I/O-Puffers
$0046: LLHH der Blocknummer
(vgl. hierzu Peeker, Heft 5/1985, S. 59, „DISKDRIVER.DEMO")
Den Vergleich der vorgegebenen *Unit-Nummer* mit der eigenen Unit-Nummer ist normalerweise entbehrlich.
Die *Blocknummer* sollte bei Bedarf mit dem zulässigen Blockbereich aufgrund der Belegung der RAM-Karte verglichen werden, um ein „Durchdrehen" beim „Low-Level"-Zugriff zu vermeiden. Weder MEGAWARP noch der 64K-Karte-Driver nehmen hierauf Rücksicht, so daß beispielsweise der 64K-Driver beim Blockread im Bereich Blocknummer $0004-$0007 nur noch „Schrott" liest.
Die Startadresse des *I/O-Puffers* wird man als korrekt annehmen müssen, da das MLI vor dem Sprung zum Driver bereits System-Bit-Map-Konflikte überprüft hat.
Von den 4 möglichen *Befehlen* sind Seek und Format auszufiltern. Bei Seek kehrt man mit CLC und Akkumulator = $00 („kein Fehler"), bei Format mit SEC und Akkumulator = $27 („I/O-Fehler") zurück. Bei Read wird der entsprechende RAM-Disk-Inhalt in den I/O-Puffer übertragen, während bei Write der I/O-Puffer-Inhalt in den entsprechenden RAM-Disk-Bereich transferiert wird. Danach wird der Driver mit CLC und Akkumulator = $00 verlassen.

Wie transferieren?

Das eigentliche Übertragen der Daten ist die Hauptaufgabe des RAM-Disk-Drivers, die jedoch nicht generell beschrieben werden kann, da sie von der Implementierung der Softswitches der Karte abhängt. Man beachte jedoch, daß vor dem Sprung zum Driver die Language-Card eingeschaltet worden ist (bei den momentanen ProDOS-Versionen Bank 1 = $C08B, bei

```
                 69   * Boot-HIMEM auf $95ØØ setzen
                 7Ø   *
1FFA: A9 95      71            LDA  #$95        ;$95ØØ
1FFC: 8D FB BE   72            STA  $BEFB
1FFF: 6Ø         73            RTS              ;$1FFF!!!
                 74            ORG  $99ØØ
                 75   *
                 76   * Megawarp-ProDOS-Driver
                 77   * ========================
                 78   *
                 79   * Originalversion von Firma IBS, Bielefeld.
                 8Ø   * Umgeschrieben und gestrafft für Driver
                 81   * außerhalb des $C8ØØ-Bereiches.
                 82   * Mit Accelerator größere Übertragungsrate.
                 83   * U.Stiehl/25.Ø5.1985
                 84   *
                 85   * Command: ØØ = Seek
                 86   *          Ø1 = Read
                 87   *          Ø2 = Write
                 88   *          Ø3 = Init
                 89   * Unit:    4Ø = Slot 4, Drive 1
                 9Ø   * Puffer:  LLHH
                 91   * Block:   LLHH
                 92   *
                 93   COMMAND  EQU  $ØØ42
                 94   UNIT     EQU  $ØØ43
                 95   PUFFERL  EQU  $ØØ44
                 96   PUFFERH  EQU  $ØØ45
                 97   BLOCKL   EQU  $ØØ46
                 98   BLOCKH   EQU  $ØØ47
                 99   *
                 1ØØ  * Es gibt 1 2Ø-Bit-Adreßregister
                 1Ø1  *         1  4-Bit-Parity-Register
                 1Ø2  *         1  8-Bit-Datenregister
                 1Ø3  *
                 1Ø4  * Datenr.     Par.      HH   MM       LL
                 1Ø5  * 7654321Ø    7654 .... 321Ø FEDCBA98 7654321Ø
                 1Ø6  * CØ83+SØ     CØ82+SØ         CØ81+SØ  CØ8Ø+SØ
                 1Ø7  *
                 1Ø8  * Das Datenregister besteht aus 2Ø Bits:
                 1Ø9  * 8-Bit-ADRLREG. = Bits Ø-7
                 11Ø  * 8-Bit-ADRMREG. = Bits 8-15
                 111  * 4-Bit-ADRHREG. = Bits 16-19
                 112  * (Die 4 restlichen Bits dienen Paritätsprüfung)
                 113  * Dies entspricht 2 hoch 2Ø = 1 Megabyte
                 114  * Die Blocknummern im Bereich $ØØØØ-$Ø7FF
                 115  * werden wie folgt umgerechnet (Beispiele:)
                 116  * Block $ØØØØ -> $ØØØØØØ HHMMLL Adreßregister
                 117  * Block $ØØØ1 -> $ØØØ2ØØ HHMMLL Adreßregister
                 118  * Block $ØØØ2 -> $ØØØ4ØØ HHMMLL Adreßregister
                 119  * Block $Ø7FF -> $ØFFEØØ HHMMLL Adreßregister
                 12Ø  *
                 121  ADRLREG  EQU  $CØ8Ø+SLOT4
                 122  ADRMREG  EQU  $CØ81+SLOT4
                 123  ADRHREG  EQU  $CØ82+SLOT4
                 124  DATAREG  EQU  $CØ83+SLOT4
                 125  *
                 126  * Blocknummer -> ADRMREG-ADRHREG
                 127  * Y-Register  -> ADRLREG
                 128  *
99ØØ: A5 46      129           LDA  BLOCKL      ;$2ØØØ!
99Ø2: ØA         13Ø           ASL
99Ø3: 8D C1 CØ   131           STA  ADRMREG
99Ø6: A5 47      132           LDA  BLOCKH
99Ø8: 2A         133           ROL
99Ø9: Ø9 1Ø      134           ORA  #%ØØØ1ØØØØ ;Parity-Bit on
99ØB: 8D C2 CØ   135           STA  ADRHREG
                 136  *
                 137  * Read/Write/Seek/Format?
                 138  *
99ØE: A5 42      139           LDA  COMMAND
991Ø: FØ 5Ø      14Ø           BEQ  OKAY        ;Seek
9912: C9 Ø3      141           CMP  #$Ø3        ;Format
9914: FØ 4F      142           BEQ  FEHLER
9916: 4A         143           LSR              ;C=1=Read
9917: A2 Ø2      144           LDX  #2          ;2 Pages
9919: AØ ØØ      145           LDY  #Ø          ;Y=Ø
991B: BØ 21      146           BCS  RDPAGEØ     ;C=1=Read
                 147  *
                 148  * Schreibvorgang
                 149  * --------------
                 15Ø  * 1. Ggf. Bit 4 von ADRH auf 1 setzen
                 151  *    für Paritätsprüfung (ORA #%ØØØ1ØØØØ)
                 152  * 2. Umgerechnete Blocknummer in
                 153  *    ADRL-, ADRM- und ADRH-Register poken
                 154  * 3. Datenbyte in Datenregister poken
                 155  *
```

späteren ProDOS-Versionen evtl. Bank 2 = $C083), so daß sich nach dem Rücksprung vom Driver der LC-Read/Write-Zustand nicht geändert haben darf. Einer der Gründe für den Read-Write-Enable-Zustand der LC liegt darin, daß sich der MLI-interne I/O-Puffer für den Directory-Block usw. in der LC befindet. RAM-Karten, die in den Bereich $D000-$FFFF „gemappt" sind (z.B. AP17 der Firma IBS), haben es hier besonders schwer. Dies gilt auch für die Übertragung eines Blocks von der LC der 64K-Karte in die LC des Motherboards. In beiden Fällen ist eine Zwischenpufferung erforderlich, die die Übertragungsrate entsprechend vermindert.

Weitere Features?

Neben den skizzierten Grundfunktionen eines RAM-Disk-Drivers sind noch weitere Features denkbar. Beispielsweise könnte man den Format-Befehl implementieren, so daß die RAM-Disk neu initialisiert werden könnte. Da jedoch das BASIC.SYSTEM den alten INIT-Befehl nicht unterstützt, wird man ihn wohl auch bei RAM-Karten nicht vorsehen müssen.

Ferner wäre es denkbar, daß man eine RAM-Disk mit PR#s bootet. Auch dieses Feature ist entbehrlich, weil unter ProDOS bereits der Reboot-Befehl – im BASIC.SYSTEM „BYE" genannt – vorgesehen ist.

Bei großen RAM-Karten ist es denkbar, daß man ähnlich wie bei Festplatten getrennte Bereiche für DOS 3.3, ProDOS, Pascal und CP/M einführt. Da jedoch der RAM-Disk-Inhalt nach dem Ausschalten des Apple gelöscht wird, dürfte sich die Mühe für eine derart diffizile Konfigurierung nur bei Spezialanwendungen lohnen.

```
991D: A5 44      156  WRPAGEØ  LDA  PUFFERL
991F: 8D 28 99   157           STA  WRPAGE1+1
9922: A5 45      158           LDA  PUFFERH
9924: 8D 29 99   159           STA  WRPAGE1+2
9927: B9 FF FF   16Ø  WRPAGE1  LDA  $FFFF,Y     ;Dummy!
992A: 8C CØ CØ   161           STY  ADRLREG
992D: 8D C3 CØ   162           STA  DATAREG     ;A->Megawarp
993Ø: C8         163           INY
9931: DØ F4      164           BNE  WRPAGE1
9933: EE 29 99   165           INC  WRPAGE1+2
9936: EE C1 CØ   166           INC  ADRMREG
9939: CA         167           DEX              ;2.Page
993A: DØ EB      168           BNE  WRPAGE1
993C: FØ 24      169           BEQ  OKAY        ;stets
                 17Ø  *
                 171  * Lesevorgang
                 172  * ----------
                 173  * 1. Umgerechnete Blocknumer in
                 174  *    ADRL-, ADRM- und ADRH-Register poken
                 175  * 2. Datenbyte aus Datenregister peeken
                 176  * 3. Ggf. Bit 4 von ADRH auf 1 prüfen
                 177  *    wegen gerader Parität (BIT ADRH)
                 178  *
993E: A5 44      179  RDPAGEØ  LDA  PUFFERL
994Ø: 8D 4F 99   18Ø           STA  RDPAGE2+1
9943: A5 45      181           LDA  PUFFERH
9945: 8D 5Ø 99   182           STA  RDPAGE2+2
9948: 8C CØ CØ   183  RDPAGE1  STY  ADRLREG
994B: AD C3 CØ   184           LDA  DATAREG     ;Megawarp->A
994E: 99 FF FF   185  RDPAGE2  STA  $FFFF,Y     ;Dummy!
                 186  *
                 187  * Wegen der Parity-Prüfung ist Blockread ca. 5Ø%
                 188  * langsamer als Blockwrite. Dies gilt jedoch nur für
                 189  * den Low-Level-MLI-Zugriff. Ansonsten ist z.B. BSAVE
                 19Ø  * langsamer als BLOAD!
                 194  *
9951: 2C C2 CØ   195           BIT  ADRHREG     ;Parity=1?
9954: 3Ø ØF      196           BMI  FEHLER
                 197  *
9956: C8         198           INY
9957: DØ EF      199           BNE  RDPAGE1
9959: EE 5Ø 99   2ØØ           INC  RDPAGE2+2
995C: EE C1 CØ   2Ø1           INC  ADRMREG
995F: CA         2Ø2           DEX              ;2.Page
996Ø: DØ E6      2Ø3           BNE  RDPAGE1
                 2Ø4  *
                 2Ø5  * Fehler?
                 2Ø6  *
9962: 8A         2Ø7  OKAY     TXA              ;X=Ø=okay
9963: 18         2Ø8           CLC              ;okay
9964: 6Ø         2Ø9  ENDE     RTS
9965: A9 27      21Ø  FEHLER   LDA  #$27        ;I/O-Error
9967: 38         211           SEC              ;nicht okay
9968: BØ FA      212           BCS  ENDE

176 Bytes
```


Für IIc und IIe mit 64K-Karte

SUPERPLOT

Double-Hires-Utility

von Karl-Walter Bott, 1984, Programmdiskette und Manual, DM 48,–

SUPERPLOT ist eine neue, ungewöhnlich kompakte und schnelle Ampersand-Utility für Double Hires, die einschließlich eines vollständigen ASCII-Shape-Zeichensatzes wahlweise in Bank 1 oder Bank 2 der Language Card liegt und damit sowohl unter ProDOS als auch unter DOS 3.3, falls letzteres in die LC-Bank geschoben wurde, benutzt und in eigene Applesoftprogramme integriert werden kann. SUPERPLOT unterstützt die üblichen HGR-Befehle, denen lediglich ein & vorangestellt werden muß, also z. B. & HPLOT 500, 100 TO 500, 150 usw. SUPERPLOT ist speziell für das Plotten von beschrifteten wissenschaftlichen Funktionskurven mit hoher Auflösung gedacht und weniger für HGR-Spiele.

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg 1

BUCH-SHOP

Betriebssystem CP/M

Vom Monitorprogramm zum Mehrbenutzersystem.
Von Jürgen Plate.
1984, 351 Seiten, 30 Abb., 3 Tab., geb., DM 56,–

Das Buch beschreibt ausführlich die Kommandos, ihre genaue Syntax und die einzelnen Teilprogramme von CP/M wie BIOS (systemspezifischer Teil), ED (Editor), ASM (Asembler, inklusive einer Beschreibung des 8080-Befehlssatzes), SYSGEN und STAT.
Der Beschreibung von CP/M ist das Listing eines komfortablen Monitorprogramms für Z-80-Computer vorangestellt, das eine elementare Programmierung auf Maschinenebene erlaubt, solange man CP/M noch nicht geladen hat. Das kann z. B. zur Fehlersuche sehr nützlich sein. Am Schluß des Buches findet sich auch eine Kurzbeschreibung der Multitasking-/Multiuser-Betriebssysteme.

Das Buch zum Apple II

von Erich Esders
1985, 210 S., 119 Abb., geb, DM 54,–

Wenn hier vom Apple II gesprochen wird, so gilt das auch für den IIplus, den IIeuroplus und die IIe-Versionen sowie für den ganzen „Apple-Nachbau". Das Buch ist ein Wegweiser durch diesen Rechner, um mit ihm schneller und effektiver zu arbeiten. Es geht hier weniger um das elementare Programmieren des Rechners, sondern um Assemblerprogramme, die extensiv Monitor-ROM-Subroutinen benutzen. Diese hat der Autor nach Sachgebieten geordnet, z. B. Mathematik, Graphik, String-Bearbeitung + Disassembler-Listings und diese wiederum mit Erklärungen und Applikationen komplettiert. Eine ausreichende Dokumentation ist dabei immer gewährleistet. Sie geht schrittweise vor, von der Aufgabenstellung über die Programmentwicklung bis zum lauffähigen Maschinenprogramm. Die angebotenen Beispiele sind ausbaufähig und lassen der eigenen Kreativität reichlichen Spielraum. Viele neuartige Tips und Tricks wird auch der beschlagene Apple-Benutzer begrüßen.
Aus dem Inhalt:
Der Mikroprozessor des APPLE II. Der APPLE II und seine Speicheraufteilung. APPLESOFT und seine Arbeitsspeicher-Bereiche. Der MICROSOFT-Basic-Interpreter: Die Zeichen-Lese-Routine. Interpretierer und Lokalisierer. Handler-Routinen. BASIC/Maschinensprache-Interfaces. DISAS-Generator. Unterprogramme im APPLESOFT-Basic-Interpreter: Softschalter und -Flags. Ausdrucks-Interpretierer. Low-Resolution-Graphik. Fehler-Behandlung. Applikationen: Arithmetik-Demonstration „FP-CALC". Hex-Dumps der Applikationen. BASIC-Monitor BASMON/D: Vorstellung der neuen Kommandos. Das Programm „BASMON/D". Implementierung und Laufbeispiele. BASIC-Interpreter-Vergleich APPLE II – Commodore 64: Arithmetik-Demonstration „FP-CALC/64". Listen: Die Token des APPLESOFT-Basic.

Apple II ROM Listing

von Matthias Buck
1984, 116 S., Kart., DM 59,–

Das deutsche Apple-II-ROM-Listing ist da!
Einleitung zum prinzipiellen Ablauf des Applesoftinterpreters:
- Aufbau und Verarbeitung der/des Programmtextes – Variablentabelle – String-space – Fließkommaformate – Basicstacks (GDSUB, FOR-NEXT, ...)
- Beschreibung der wichtigsten Unterprogramme, z. B. Variablensuche, Garbage collection, Ausdrucksauswertung, CHRGET, ...
- Vollständig disassemblierte und sehr ausführlich deutsch kommentierte Auflistung des Applesoft-BASIC-Interpreters
- Übersichtliche Auflistung aller vom Interpreter benutzten RAM-Zeilen mit allen Verwendungszwecken
- Über 150 ausführlich dokumentierte Unterprogramme:
  – Funktion
  – Ein/Ausgabeparameter

Auch für Apple-IIe und c und Kompatible!

Apple II Pascal

Eine praktische Anleitung
von Arthur Luehrmann und Herbert Peckham
1982, 544 S., kart., DM 59,–

Dieses Buch ist unentbehrlich für alle, die die Programmiersprache PASCAL lernen wollen und Zugang zu einem Apple Computer haben.
Sie benötigen keinerlei Vorkenntnisse, sondern lernen an Hand von Beispielen und Übungen, wie man selbst PASCAL-Programme entwikkelt und sie austestet und werden allmählich von Kapitel zu Kapitel vertrauter im Umgang mit dem Apple Computer.

Start mit Apple-Logo für II, IIe und IIc

Das kleine Logo-Einmaleins: Grafik * Text * Musik
Von D. Senftleben
1985, 222 Seiten, DM 35,–

Viele Mikrocomputer-Hersteller bieten für ihre Geräte neben BASIC und anderen Programmiersprachen zunehmend auch Logo an. Durch ihre Benutzerfreundlichkeit hat diese Sprache bereits viele Freunde im Ausbildungs- und Freizeitbereich gefunden. Dabei ist Logo eine mächtige Sprache, die auch dem anspruchsvollen Anwender kaum Wünsche offenläßt. Mittels Schildkrötengrafik wird das kleine Logo-Einmaleins in 12 Lektionen entwikkelt. Große Bildschirmfotos begleiten den Leser durch die Lektionen. Das Buch verlangt aktive Mitarbeit. Es hat seinen Platz neben dem Computer und gibt Hilfen und Anregungen für eigenes Forschen. Dank des bausteinorientierten Konzepts kann jeder seine eigenen Teilbausteine erzeugen und sie zu neuen Blöcken zusammenfügen. Neben dem Einmaleins werden neue Einsatzbereiche für den Einsteiger erschlossen. Musik und Sound fehlen nicht.
In diesem Buch werden die beiden offiziellen Logo-Produkte der Firma Apple für die Rechnerfamilie Apple II, IIe und IIc behandelt und deren Unterschiede verdeutlicht. Weiterhin sind sämtliche Apple-Logo-Vokabeln übersichtlich zusammengestellt. Dieses Buch ist ideal zum problemlosen und vergnüglichen Start in die Apple-Logo-Welt.

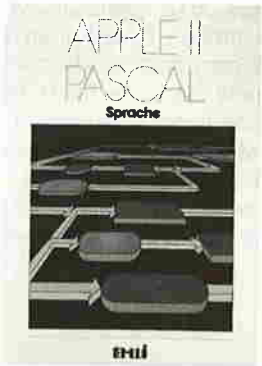
Apple II Anwenderhandbuch

von Lon Poole
1982, 450 S., zahlr. Abb., kart., DM 56,–

Auch für diesen Computer haben wir den richtigen Leitfaden. Er erspart Ihnen zeitraubendes und nutzloses Suchen nach der wirklich verwendbaren Dokumentation für Ihren Computer. Das Anwenderhandbuch beschreibt zum einen den beliebten Apple II-Computer als solchen und gibt zum anderen ausführlich Auskunft über die normalen Peripheriebausteine und Zubehör einschließlich Disk-Laufwerken und Drucker. Mit Hilfe dieses Buches werden Sie Ihren Apple II erfolgreich einsetzen, denn der Informationsgehalt geht weit über das hinaus, was herstellerseitig an Literatur angeboten wird. Sie lernen BASIC auf zwei verschiedene Arten zu verwenden. Wie man den Gebrauch von Klang, Farbe und Grafik zum Optimum führt. Sie erhalten Tips für fortgeschrittene Programmerstellung. Sie erfahren die Verwendung des Maschinensprachen-Monitors u. v. m. Mit dem Apple II-Anwenderhandbuch werden Ihnen alle Möglichkeiten eröffnet, die in diesem Computer stecken.

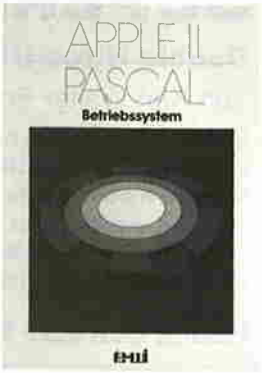
Apple II Pascal

Sprache
1985, 197 S., DM 39,–



Apple II Pascal

Betriebssystem
1985, 256 S., DM 49,–



BUCH-SHOP

Apple DOS 3.3

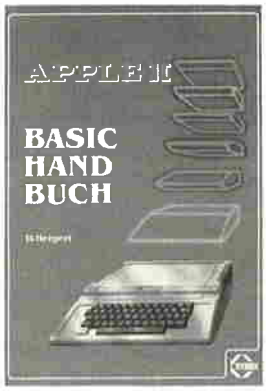
von Ulrich Stiehl
2. Aufl. 1984, 203 S., kart., DM 28,–

Dies ist die erste deutschsprachige Darstellung des Diskettenbetriebssystems DOS 3.3 für den Apple II/II Plus/IIe, die sich sowohl an Applesoft- als auch an Assembler-Programmierer wendet. Sinngemäß ist das Buch zweigeteilt:
Der erste Teil behandelt ausführlich die dem Applesoft-Programmierer zur Verfügung stehenden DOS-Befehle, wobei die Textfiles wegen ihrer großen Bedeutung und der vergleichsweise komplizierten Handhabung besonders dargestellt werden. Viele Textfile-Tricks werden hier zum erstenmal geschildert.
Aber auch im zweiten Teil findet der reine Applesoft-Programmierer insbesondere in dem Kapitel „Vermischte Tips, Tricks und Patches" zahlreiche Anregungen. Im übrigen ist der zweite Teil für Assembler-Programmierer gedacht. Neben einer detaillierten Beschreibung der DOS-Interna enthält dieser Teil elf vollständige RWTS-Anwenderprogramme – z. B. CPM-Refiner, DOS-lose Datendisk, TSL-Maker, File-Reader, Pseudo-Disk-Driver und Fastbrun-Routine –, die Techniken enthüllen, die bislang noch niemals publiziert worden sind. Dieses DOS-Buch ist deshalb der unentbehrliche Begleiter für jeden Apple-Programmierer.

Apple II Basic Handbuch

von Douglas Hergert
304 Seiten, 116 Abb.
DM 32,–

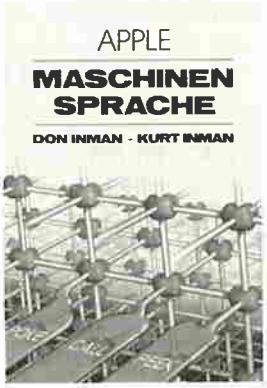
Das Buch ist als Nachschlagewerk konzipiert, daß seinen Platz neben jedem APPLE II, II+ und IIe haben sollte. Es richtet sich an Anfänger und fortgeschrittene Programmierer.



Aus der Praxis heraus präsentiert der Autor Tips und Vorschläge, die das Programmieren leichter und zugleich effizienter machen. Alle Applesoft- und Integer-BASIC-Begriffe sind alphabetisch aufgelistet und werden eingehend erklärt. Dazu werden alle DOS-Befehle (neben vielen Begriffen der Computerterminologie) vorgestellt.
Beispielprogramme zeigen dem Nutzer, wie jeder Befehl funktioniert und helfen, die richtige Anwendung zu üben. Unter anderem lernt der Leser den besten Weg, um FOR/NEXT-Schleifen und IF/THEN-Entscheidungen für seine Zwecke einzusetzen.
Durch die präzise und leicht verständliche Sprache des Autors werden auch schwierige Befehle einfach in der Anwendung.

Apple Maschinensprache

von Don und Kurt Inman
1984, 208 S., zahlr. Abb. und Tabellen, DM 49,–



Dieses Buch ist wahrscheinlich die beste Einführung in die 6502-Programmierung für denjenigen Assembler-Anfänger, der zuvor noch nie ein Maschinenprogramm geschrieben hat.
Aus dem Inhalt:
Applesoft II BASIC – kurzgefaßt – Alles über Zeichen – Alles über Speicher – Alles über Maschinenbefehle – Maschinenprogramme mit BASIC eingeben – Graphik – Text – Ton – Arithmetik – Was tun mit den Maschinenprogrammen

Apple II leicht gemacht

von Joseph Kascmer
1984, 185 S., zahlr. Abb., kart., DM 28,–

Dies ist ein Buch, wie es sich jeder Apple-Anfänger nur wünschen kann: Schrittweise, leichtverständliche Anleitung zum Umgang mit dem Apple mit einigen durchsichtigen, unkomplizierten Beispielen in Applesoft, die ihn nicht Abschrekken, sondern ermutigen sollen, sich mit dem Gerät näher vertraut zu machen. Damit ist „Apple II leicht gemacht" das ideale Einsteigerbuch für den reinen Anwender, der nicht nur „auf den Knopf drücken", sondern zumindest einige Details aus der Black Box namens Apple erfahren will.

Aus dem Inhalt:
Kontrolle des Geräts – Schreiben und Zeichnen auf dem Bildschirm – Geheimnisvolle Abläufe: Programme – Verschiedene Eingriffsmöglichkeiten – Mobile Speicher: Disketten – Kontrollmöglichkeiten – Das Innenleben

Apple Assembler

Tips und Tricks
von Ulrich Stiehl
1984, 226 S., 3 Abb., kart., DM 34,–

„Apple Assembler" wendet sich an alle, die bereits Anfängerkenntnisse der 6502-Programmierung haben – z. B. aufgrund des Buches „Apple Maschinensprache" – und nunmehr ein Nachschlagewerk für ihren Apple II Plus/IIe/IIc suchen, in dem alle wichtigen ROM-Routinen sowie eine Vielzahl sonstiger Hilfsprogramme in einer systematischen Form zusammengestellt werden. Insgesamt umfaßt dieses Buch über 40 Utilities, darunter mehrere völlig neuartige Programme wie Double-Lores, Double Hires, Screen-Format u. a.
Der erste Teil enthält ein Repetitorium der wichtigsten Befehle, Adressierungsarten und sonstigen Besonderheiten des 6502.
Im zweiten Teil werden alle Adressen des Monitors zusammengestellt, die für Assembler-Programmierer von Nutzen sein können. Darüber hinaus findet der Leser Unterroutinen für hexadezimale Addition/Subtraktion/Multiplikation/Division, Binär-Hex-ASCII-Umwandlung usw.
Der dritte Teil befaßt sich mit der Speicherverwaltung der Language Card und der IIe-64K-Karte und enthält Move-Programme zum Verschieben von Daten in die und aus der Language Card sowie der 64K-Karte.
Der vierte Teil ist dem Applesoft-ROM gewidmet und listet eine große Anzahl nützlicher Interpreter-Adressen. Bei den Utility-Programmen liegt das Schwergewicht auf Fließkommamathematik einschließlich Print Using.
Der letzte Teil behandelt den Text- und Graphikspeicher. Neben einem professionellen Maskengeneratorprogramm werden auch Routinen zur Double-Lores- und Double-Hires-Grafik vorgestellt.

Arbeiten mit dem Macintosh

von N. Hesselmann
416 Seiten, 320 Abb. DM 54,–

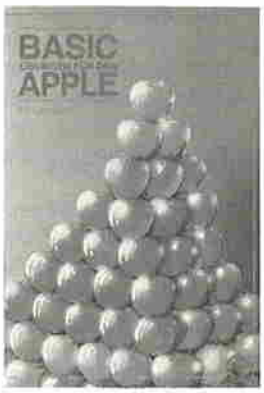
Das Buch erklärt den Umgang mit dem Macintosh von Grund auf, wobei auch auf elementare Dinge eingegangen wird, wie z. B. die Benutzung der Tastatur und der Maus, das Einlegen von Disketten und den Systemstart. Ganz besonderes Augenmerk wird auf die Erklärung der speziellen Software-Umgebung des Macintosh gelegt, wobei das Menü- und Fensterkonzept sowie das Anwählen durch Piktogramme gekennzeichneter Funktionen klar dargestellt wird.

Der Umgang mit den Programmen MacPaint und MacWrite wird erläutert; dies geschieht teilweise anhand von Beispielen, die leicht nachvollzogen werden können. Ein umfangreiches Kapitel ist dem für den Macintosh erhältlichen Microsoft-BASIC gewidmet.

BASIC Übungen für den Apple

von J. P. Lamoitier
1983, 252 S., zahlr. Abb., kart., DM 38,–

Das Buch ist konzipiert, allen Apple-Anwendern Applesoft-BASIC durch praktische Übungen an Hand von reellen Programmen beizubringen. Datenverarbeitung, Statistik, kommerzielle Programme, Spiele und vieles mehr. Jede Übung beinhaltet eine Beschreibung der Problemstellung, eine Analyse der Lösungsmöglichkeiten, ein Flußdiagramm und ein fertiges Programm samt Probelauf.



Aus dem Inhalt:
Ihr erstes BASIC-Programm – Flußdiagramme – Übungen mit Integerzahlen – Elementare Beispiele aus der Geometrie – Allgemeine Übungen aus der Datenverarbeitung – Mathematische Berechnungen – Kaufmännische Berechnungen – Spiele – Operations Research – Statistik

Apple ProDOS für Aufsteiger

Band 1
von Ulrich Stiehl
1984, 202 S., kart., DM 28,–

ProDOS ist das neue „professionelle DOS" (Professional Disk Operating System) für den Apple IIe sowie den mit einer Language Card ausgestatteten Apple II Plus. Band 1 befaßt sich mit den theoretischen Grundlagen von ProDOS, der internen und externen Speicherorganisation und enthält grundlegende Beispielprogramme für Assembler-Programmierer sowie generelle Untersuchungen zum BASIC-SYSTEM. Da ProDOS über erheblich vielfältigere und leistungsfähigere, zugleich jedoch erheblich kompliziertere Dateistrukturen verfügt, sind theoretische Kenntnisse von ProDOS unabdingbar, wenn man die Features von ProDOS voll ausschöpfen will.
Aus dem Inhalt:
Ein erster Überblick – ProDOS und DOS 3.3 – Interne Speicherorganisation – Externe Speicherorganisation – MLI (Machine Language Interface) – ProDOS für Applesoft-Programmierer

Beachten Sie die Buch-Shop-Karte

FORMATTER

Ein universelles Formatierungsprogramm

von Arne Schäpers

Ursprünglich war dieser Artikel nebst Programm als Teil einer weitergehenden Beschreibung der Apple-Drives vorgesehen. Um aber allzugroße Überschneidungen mit der Artikelserie von Gerhard Berg zu vermeiden, bringen wir dieses Programm als „Stand-alone"-Utility mit einem Minimum an Erklärungen.

Die drei „kritischen" Punkte beim Lesen von und Schreiben auf Diskette, nämlich
a) Nibble-Kodierung,
b) die zeitlichen Bedingungen beim Schreiben eines Bytes und
c) Bewegungen des Arms
werden in einer späteren Ausgabe vom Peeker ausführlich behandelt.

Eine Begründung für das folgende Programm sollte hier auch nicht fehlen:
So gut wie alle Disk-Utilities sind im besten Fall für vollständig kompatible 40-Track-Drives geeignet – Benutzern von Drives mit mehr als 40 Tracks bleibt nur die Hoffnung, daß nichts schiefgeht.

Aus diesem Grund wird in einer der nächsten Peeker-Hefte auch ein Disk-Editor für ProDOS (und DOS 3.3) erscheinen, der sich streng an den Standard hält mit der Folge, daß alle mit Hilfe von „Patches" angepaßten Drives einschließlich der RAM-Disk benutzt werden können.

Das hier vorgestellte Formatierungsprogramm hat folgende Features:
– Es arbeitet mit 35 bis 80 Tracks in jedem Slot (außer 0);
– Die Neuformatierung einzelner Tracks ist möglich;
– Es kann ermittelt werden, wieviele Tracks ein Drive „verträgt";
– Es läuft unter DOS 3.3 und unter ProDOS.
Man beachte jedoch, daß nur eine Formatierung im engeren Sinne vorgenommen wird, d.h. es werden weder Boot- und Directory-Blocks (bei Pascal- und ProDOS) noch Catalog-Spuren (bei DOS 3.3) angelegt. Das Formatierungsprogramm kann deshalb um eine Blockwrite- (Pascal, ProDOS) oder RWTS-Routine erweitert werden, die die bei dem jeweiligen Betriebssystem erforderlichen System-Blocks oder -Spuren anlegt.

Die Theorie

Zur Einleitung der Theorie zunächst einmal die Speicherstellen (I/O-Adressen) des Controllers in tabellarischer Form:
$C080–$C087: Phasen des Schrittmotors zur Armsteuerung
$C088: Drive-Motor aus
$C089: Drive-Motor an
$C08A: Selektierung Drive 1
$C08B: Selektierung Drive 2
$C08C: Strobe des Data Latch
$C08D: Load des Data Latch
$C08E: Setzt „Read" von der Diskette
$C08F: Setzt „Write" auf die Diskette
Alle diese Speicherstellen sind hier für Slot 0 angegeben. Bei einem anderen Slot kommt noch die Slotnummer * 16 hinzu, also z.B. für das Anschalten eines Drives in Slot 6:
LDX #$60 (= Slotnummer * 16)
LDA $C089,X (= Motor an)

Über das Ansprechen von Drive 1 oder Drive 2 entscheiden nur die Speicherstellen $C08A und $C08B. Alle folgenden Routinen finden mit dem durch diese beiden Speicherstellen selektierten Laufwerk statt.
Mit Ausnahme der Speicherstellen $C08D bis $C08F ist das Ansprechen durch einen beliebigen Prozessor-Befehl, also z.B. LDA oder LDX, aber auch CMP oder BIT ausreichend.

Ein Formatierungsprogramm besteht im allgemeinen aus vier Teilen:

Parameterübergabe
Zunächst müssen die Werte wie Start-Track, End-Track, Volume-Nummer usw. festgelegt werden. Diese Aufgabe wird durch das Applesoft-Programm namens **FORMAT** erledigt, das hier aus rein optischen Gründen etwas „professioneller" ausgefallen ist.

Selektierung und Anschalten des Ziel-Drives
Bei einem „richtigen" DOS fällt dieser Prozeß wesentlich komplizierter aus. So wird z.B. zuerst geprüft, ob noch ein anderer Drive an ist usw. Unser Formatierungsprogramm beschränkt sich auf das Ansprechen der Speicherstelle $C089 (+ Slotnummer * 16) sowie die Selektierung von Drive 1 oder Drive 2. Danach folgt eine Verzögerung, damit sichergestellt wird, daß der Motor die Nenndrehzahl erreicht. Zweckmäßigerweise wird innerhalb dieser Zeit der Arm rekalibriert.

Schreiben eines Tracks
Zum „normalen" Schreiben eines Sektors (Blocks) bestehen hier mehrere gravierende Unterschiede:
– Es wird „blind" geschrieben, d.h. es kann naturgemäß vorher nicht durch Lesen eines Address-Field geprüft werden, ob sich der Arm auf dem richtigen Track befindet. Der Arm wird einfach pro Track um eine Position nach innen bewegt.
– Es muß nicht nur ein Data-Field, sondern auch jeweils vorher ein entsprechendes Address-Field geschrieben werden.
– Das Data-Field bleibt leer und enthält nur Nullen ($96).

Verifizierung des geschriebenen Tracks
Nach dem Schreiben wird der Track nochmals gelesen, um etwaige Übertragungsfehler oder Diskettenschäden festzustellen.

Der Aufbau einer Spur und die Einteilung eines Sektors in verschiedene Felder kann der **Tabelle 1** entnommen werden.

Das Programm

Das Applesoft-Programm **FORMAT** wurde geschrieben, um die Parameterübergabe etwas komfortabler zu gestalten. Das Setzen der einzelnen Parameter könnte ebenso durch diverse Pokes und ein anschließendes BRUN von FORMAT.OBJ stattfinden. Diese Werte werden in der Page 3 übergeben und sind in **Tabelle 2** aufgeführt.

Das Maschinenprogramm **FORMAT.OBJ** hat vier Einsprünge:

8192 ($2000) – Drive Select, Motor an, Rekalibrierung auf Track 00;
8195 ($2003) – Motor an;
8198 ($2006) – Motor aus;
8201 ($2009) – Arm auf Track xx, und Formatierung von Track xx bis Track yy mit anschließender Verifizierung. Das fast schon übliche Knirschgeräusch des Arms nach beendeter Formatierung entfällt, weil die Formatierung in aufsteigender, die Verifizierung dagegen in absteigender Richtung vorgenommen wird, d.h. bei einem kompletten Durchgang befindet sich der Arm danach wieder über Track 00.

Eine abschließende Warnung, falls Sie FORMAT.OBJ modifizieren wollen: Die Routine WRTRACK (ab $22BB) schreibt einen ganzen Track synchronisiert, d.h. in einem auf die Mikrosekunde abgestimmten Zeitverhältnis. Bereits das speicherplatzmäßige Verschieben kann u.U. zu unlesbarem Müll auf der Diskette führen, wenn z.B. danach ein relativer Sprung (BNE ..) über eine Seitengrenze hinausgeht, denn der 6502 hat leider die unangenehme Eigenschaft, sich für derartige Sprünge eine Mikrosekunde mehr Zeit zu lassen.

Tabelle 1: Aufbau einer Spur

```
GAP 1:          eine größere Anzahl von Autosyncs ($FF)
                                  :
                                  :
                                  :
Address-Field:  Address-Field-Header ($D5 $AA $96)
                Volume-Nummer (1 Byte i.allg. $FE)
                Track-Nummer (1 Byte)
                Sektor-Nummer (1 Byte)
                Checksum (2 Bytes)
                Address-Field-Trailer ($D5 $AA $EB)

GAP 2:          mehrere Autosyncs ($FF)

Data-Field:     Data-Field-Header ($D5 $AA $AD)
                Information des Datensektors (341 Bytes)
                Checksum (2 Bytes)
                Data-Field-Trailer ($D5 $AA $EB)

GAP 3:          mehrere Autosyncs ($FF) bis zum Anfang des
                Address-Fields des nächsten Sektors
                                  :
                                  :
                                  :
Address-Field
GAP 2                (weitere Sektoren)
Data-Field
                                  :
                                  :
                                  :
GAP 3 des Sektors $ØF
GAP 1
```

Das GAP 3 des Sektors $ØF, also des letzten Sektors auf dem Track, reicht dann bis in das GAP 1 hinein, d.h. GAP 1 wird teilweise überschrieben. Somit ist gewährleistet, daß der gesamte Track definierte Information enthält.

Tabelle 2: Übergabeparameter

```
$Ø3ØØ (768): Slot-Nummer * 16 (z.B. 96 für Slot 6)
$Ø3Ø1 (769): Drive-Nummer (Ø oder 1)
$Ø3Ø2 (77Ø): Volume-Nummer (z.B. $FE)
$Ø3Ø3 (771): Start-Track (z.B. Ø)
$Ø3Ø4 (772): End-Track (z.B. 34)
$Ø3Ø5 (773): Half-Track ($ØØ = ganze Spur,
                         $8Ø = Halbspur)
```

FORMAT

```
1ØØ TEXT : HOME : INVERSE : PRINT SPC( 4Ø): NORMAL :
    VTAB 1: HTAB 15: PRINT " FORMATTER "
1Ø5 REM
11Ø IF PEEK (8192) = 4 * 16 + 12 THEN 13Ø
115 VTAB 6: PRINT "FORMAT.OBJ wird geladen..."
12Ø PRINT CHR$ (4)"BLOAD FORMAT.OBJ"
125 REM **** Initialisierung und Parameter-Anzeige ****
13Ø LOMEM: 1Ø24Ø: REM Damit FORMAT auch unter ProDOS
    läuft!
135 SL$ = "6":DR$ = "2":VL$ = "FE": REM Slot, Drive,
    Volume
14Ø ST$ = "ØØ":ET$ = "34":HT$ = "N": REM Anfang, Ende,
    Half-Track
145 POKE 34,1: HOME
15Ø POKE 33,34: POKE 32,4: REM Textfenster
155 VTAB 4: PRINT "Slot: ";SL$;"   Drive: ";DR$;"
    Volume: $"VL$
16Ø VTAB 6: PRINT "Start-Track: ";ST$;"    End-Track:
    ";ET$
165 VTAB 8: PRINT "Software "; CHR$ (34);"Half-Track";
    CHR$ (34);" (J/N): ";HT$
17Ø POKE 32,Ø: POKE 33,4Ø
175 REM **** Restart ****
18Ø VTAB 12: HTAB 8: PRINT "  <P> = andere Parameter  "
185 VTAB 14: HTAB 6: PRINT "<CR> = Start     <ESC> =
    Ende"
19Ø REM **** Parameter/Start/Ende ****
195 VTAB 14: HTAB 2Ø: GET X$
2ØØ IF X$ = CHR$ (27) THEN HOME : POKE 34,Ø: END
2Ø5 IF X$ = CHR$ (13) THEN 46Ø: REM Start
21Ø IF X$ < > "P" AND X$ < > "p" THEN 195
215 REM **** Parameter ändern ****
22Ø POKE 34,11: HOME : POKE 34,1
225 RESTORE :N3$ = "":N4$ = ""
23Ø A$ = SL$: GOSUB 3Ø5:SL$ = A$
235 A$ = DR$: GOSUB 3Ø5:DR$ = A$
24Ø N3$ = "A":N4$ = "F": REM Hexzahlen erlaubt
245 A$ = VL$: GOSUB 33Ø:VL$ = A$
25Ø N3$ = "":N4$ = "": REM Nur Dezimalzahlen
255 A$ = ST$: GOSUB 33Ø:ST$ = A$
26Ø A$ = ET$: GOSUB 33Ø:ET$ = A$
265 IF VAL (ST$) < = VAL (ET$) THEN 275
27Ø VTAB 12: HTAB 5: PRINT CHR$ (7);"Start-Track größer
    End-Track ??": GOTO 225
275 IF VAL (ET$) - VAL (ST$) < 4Ø THEN HT$ = "N": GOTO
    285
28Ø A$ = HT$:N3$ = "Y":N4$ = N3$: GOSUB 3Ø5:HT$ = A$
285 VTAB 8: HTAB 34: PRINT HT$
29Ø POKE 34,11: HOME : POKE 34,1
295 GOTO 18Ø
3ØØ REM *** Subroutinen für Parameteränderungen ***
3Ø5 REM Input ein Zeichen
31Ø GOSUB 38Ø: REM Setzt Cursor
315 GET X$: IF X$ = CHR$ (13) THEN X$ = A$: RETURN
32Ø GOSUB 4ØØ: IF X$ = "" THEN 315: REM Check
325 A$ = X$: PRINT A$;: RETURN
33Ø REM Input zwei Zeichen
335 GOSUB 38Ø: REM Setzt Cursor etc.
34Ø GET X$: IF X$ = CHR$ (13) THEN X$ = A$: RETURN
345 GOSUB 4ØØ: IF X$ = "" THEN 34Ø: REM Check
35Ø Z$ = X$: PRINT X$;" "; CHR$ (8);
355 GET X$: IF X$ = CHR$ (13) THEN A$ = "Ø" + Z$: GOTO
    375
36Ø IF X$ = CHR$ (8) THEN HTAB H: PRINT A$;: HTAB H:
    GOTO 34Ø: REM Restart
365 GOSUB 4ØØ: IF X$ = "" THEN 355
37Ø A$ = Z$ + X$
375 HTAB H: PRINT A$: RETURN
38Ø REM Setzt Cursor und liest Min/Max
385 READ V,H: VTAB V: HTAB H
39Ø READ N1$,N2$: REM Min/Max
395 RETURN
4ØØ REM UC-Übersetzung und Check
4Ø5 X$ = CHR$ ( ASC (X$) - 32 * ( ASC (X$) > 96))
41Ø IF (X$ < N1$ OR X$ > N2$) AND (X$ < N3$ OR X$ > N4$)
    THEN PRINT CHR$ (7);:X$ = ""
415 RETURN
42Ø REM VTABs, HTABs und Min/Max für Parameter
```

```
425 DATA 4,11,"1","7": REM Slot
430 DATA 4,22,"1","2": REM Drive
435 DATA 4,35,"0","9": REM Volume
440 DATA 6,18,"0","9": REM Start-Track
445 DATA 6,35,"0","9": REM End-Track
450 DATA 8,34,"N","N": REM Half-Track, "Y" ist N3$/N4$
455 REM **** Start FORMAT ****
460 POKE 34,11: HOME : POKE 34,1: VTAB 12: HTAB 8
465 PRINT "<ESC> = Funktion abbrechen"
470 POKE 768,16 * VAL (SL$): POKE 769, VAL (DR$) - 1
475 VL = 0: FOR X = 1 TO 2:X$ = MID$ (VL$,X,1)
480 VL = 16 * VL + ASC (X$) - 48 - 7 * ( ASC (X$) > 64):
    NEXT
485 POKE 770,VL: POKE 771, VAL (ST$): POKE 772, VAL (ET$)
490 POKE 773,0: IF HT$ = "J" THEN POKE 773,127
495 REM 8192 = Drive on, Recal., Test Track 00
500 REM 8195 = Drive on
505 REM 8198 = Drive off
510 REM 8201 = FORMAT (Start..End-Track)
515 VTAB 16: HTAB 1
520 PRINT "0123456789ABCDEF0123456789ABCDEF01234567";
525 PRINT "----------------------------------------"
530 VTAB 18: HTAB 1: CALL 8192
535 ERR = PEEK (767)
540 IF ERR < > 0 AND ST$ = "00" THEN 630: REM Disk neu
    => FORMAT
545 IF ERR = 0 AND ST$ < > "00" THEN 630: REM Disk
    teilweise formatiert
550 CALL 8198: REM Drive aus"
555 VTAB 21: PRINT CHR$ (7)
560 IF ERR = 0 THEN 585: REM Disk nicht neu
565 PRINT "Track 00 unformatiert -"
570 PRINT
575 PRINT "Positionierung auf Track "ST$" unmöglich."
580 GOTO 180
585 VL = PEEK (768 + 9):VL$ = "": FOR X = 1 TO 2
590 VL$ = VL$ + CHR$ ((VL / 16) + 48 + 7 * (VL / 16 >
    10))
595 VL = (VL - INT (VL / 16) * 16) * 16: NEXT
600 VTAB 4: HTAB 35: PRINT VL$: VTAB 22
605 PRINT "Diskette enthält Daten!"
610 PRINT
615 PRINT "Neu formatieren (J/N): ";
620 GET X$: IF X$ < > "J" AND X$ < > "j" THEN PRINT
    "N";: GOTO 180
625 PRINT X$;: CALL 8195: REM Drive an
630 POKE 34,20: HOME : POKE 34,1
635 VTAB 22: PRINT "Disk wird formatiert..."
640 VTAB 18: HTAB 1: CALL 8201
645 VTAB 22: PRINT SPC( 30);: HTAB 1
650 IF PEEK (767) = 0 THEN PRINT "Formatierung
    fehlerfrei beendet.": GOTO 180
655 PRINT CHR$ (7);: ON PEEK (767) GOTO
    660,680,690,710,730
660 PRINT "Kein Laufwerk angeschlossen oder"
665 PRINT
670 PRINT "Diskette schreibgeschützt.";
675 GOTO 180
680 PRINT "Laufwerksgeschwindigkeit zu hoch."
685 GOTO 180
690 PRINT "Allgemeiner Fehler: Schlechtes Medium,"
695 PRINT
700 PRINT "Klappe offen, etc.";
705 GOTO 180
710 PRINT "Armpositionierungsfehler -"
715 PRINT
720 PRINT "wahrscheinlich zu hohe Track-Nummer.";
725 GOTO 180
730 PRINT "- Abbruch."
735 GOTO 180
740 REM Arne Schäpers 4/85
```

Diverse Experimente mit Original-Disk-II-Drives und Duodisk-Drives (A. Schäpers benutzt andere Laufwerke) zeigten, daß bei mit FORMAT initialisierten DOS-3.3- und ProDOS-Disketten beim RWTS-Sektor-Write bzw. MLI-Block-Write der **Zugriff doppelt solange dauert** wie beim Read. Aus Zeitgründen war es mir bislang nicht möglich, die optimalen Werte für die Synchronisationsbytes zu finden. *us*

ZUSATZ-KARTEN:

V-24-Schnittstelle 199,- **Z-80-Karte** 139,-
80-Zeichen-Karte m.Softswitch 236,- **16 K-Language-Karte** 138,-
Centronics-Karte von Epson für Graphik .... 210,- für Text 145,-
Centronics-Schnittstelle für 2 Drucker gleichzeitig 129,-
Eprommer incl. Software 198,-

Super-Eprommer
belegt keinen Slot, incl. Software für 2708-27128 ..... **239,-**
128-K-Karte ..... auf Anfrage

Floppy-Controller
FDC 4 für alle Laufwerke ..... 170,- Bausatz wie links 159,-
Leerplatine wie oben incl. Prom u. Eprom 98,-
Druck Buffer 64 K ..... auf Anfrage

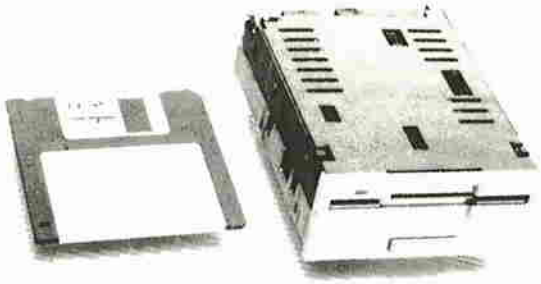
Autopatch-Controller (Erphi Controller)
1 x 35 bis 2 x 80 Tr.-Disk, keine Patch-Disk notwendig, Patch DOS 3.3, Diversi-DOS 2-C, 4-C (DD MOVER), Pascal 1.1, Pascal 1.2, CPM 2.2, ProDOS 1.0.1, 1.1, 1.1.1; Sie können die Laufwerke untereinander mischen .. **298,-**

Joy Stick De Luxe .. **59,-** **Netzteil 5A** **149,-**

Gehäuse für 1 5¼" Slimline Laufwerk 39,-
Gehäuse für 2 5¼" Slimline Laufwerke mit Platz für ein Netzteil 159,-
Gehäuse für 2 3½" Slimline Laufwerke mit Platz für ein Netzteil 79,-
IBM®-Gehäuse 229,-
Floppy-Kabel 34pol. für 2 Laufwerke mit Shugart-Bus 35,-

Preh Commander Keyboards
Wir bieten Ihnen die **Preh-Qualität** auch für Apple. AK 88 Spez. mit Gehäuse, Anschlußkabel, Zehner-Tastenfeld, dt. Zeichensatz, Sondertasten für Ctrl-Codes und Rechenfunktionen **339,-**
Preh Commander Keyboard AK 87, frei programmierbar bis zu 10 Ebenen, pro Taste bis zu 250 Zeichen ..... nur **559,-**

TEAC 3½" Laufwerk FD 35 F 535,-
Speicherkapazität 1 MB, (formatiert 640 KB) jetzt für nur .....



TEAC FD 55 A 1 x 40 Track ..... 445,- **TEAC FD 55 B** 2 x 40 Track 515,-
TEAC FD 55 E 1 x 80 Track ..... 490,- **TEAC FD 55 F** 2 x 80 Track 560,-

Philipps Slimline Floppy X3134 A, 2x80 Track solange Vorrat ..... nur 569,-
SONY 3½" Laufwerk ..... nur 799,-
Apple®-kompatibles Laufwerk incl. Gehäuse + Kabel 599,-
320 KB Laufwerk für IIc **948,-**
148 KB Laufwerk für IIc **698,-**

EPSON DRUCKER
EPSON FX 80 ..... 1670,- **EPSON FX 100** 2159,-
EPSON RX 80 ..... 1079,- **EPSON RX 80 FT** 1295,-

Patch-Diskette für **SONY 3½" Laufwerke** – ermöglicht die Anpassung an II/IIe und kompatible Computer ..... 80,- Manual vorab 15,-DM (wird beim Kauf der Patch-Diskette angerechnet).
dto. für 5¼" Laufwerke 69,-
Disketten Döb&Böd SS/DD 10St. 62,- **10 Disketten** f. 3½" Laufw. 150,-
Akustikkoppler AK 300 mit FTZ-Nr. incl. Netzteil 549,-
Interface zum Anschluß von Laufwerken mit Shugart-Bus 248,-

Die Microfloppy mit Zukunft:
Speicherkapazität: 2 x 1 MByte formatiert: 2 x 640 kByte. Anschlußfertig mit PROM-residenter Patchsoftware für CP/M 2.2, Apple DOS 3.3, Diversi-DOS 2-C, 4-C (DD MOVER), Apple Pascal 1.1, Pascal 1.2, Pro-DOS 1.0.1, 1.1, 1.1.1 zum Preis von **1640,-**
Low Power Version 1740,-

10 MB Winchester
mit Software für DOS 3.3, CP/M 2.20, Pascal, Pro-DOS, incl. Controler und Gehäuse **4490,-**

Sonderangebot
Disketten Döb&Böd 40 Track, SS/DD 10er Pack ... **49,-**

Gesamt-Preisliste anfordern! **Preise inclusive gesetzlicher Mehrwertsteuer.**

UEDING electronics
Holtewiese 2
5750 Menden 1
DFÜ 02373/66877
Tel. 02373/63159

FORMAT.OBJ

```
                  1   *
                  2   * vom Steuerprogramm gesetzte Werte
                  3   *
                  4   SLOT16   EQU  $300      ; Slot-Nr. * 16
                  5   DRIVENO  EQU  $301      ; Drive: 0 oder 1
                  6   VOLNO    EQU  $302      ; Volume-Nummer
                  7   STRACK   EQU  $303      ; Start-Track
                  8   ETRACK   EQU  $304      ; End-Track
                  9   HFLAG    EQU  $305      ; "Half-Track" Flag: $00 oder $80
                  10  *
                  11  ERRNO    EQU  $2FF      ; Fehlernummer für BASIC
                  12  *
                  13  * vom Programm benutzte Speicherstellen
                  14  *
                  15  ADDRBASE EQU  $306      ; Basis der von RDADDR
                  16  SECFOUND EQU  $307      ; gesetzten Speicherstellen:
                  17  TRKFOUND EQU  $308      ; Volume/Track/Sektor
                  18  VOLFOUND EQU  $309      ; und CHECKSUM in absteig. Folge
                  19  *
                  20  TRNOW    EQU  $30A      ; "Track jetzt"
                  21  TRWANT   EQU  $30B      ; "Track gewünscht"
                  22  *
                  23  * Speicherstellen des Controllers
                  24  *
                  25  ARMBASE  EQU  $C080     ; Phasen-Adressierung
                  26  DRVOFF   EQU  $C088     ; Motor off
                  27  DRVON    EQU  $C089     ; Motor on
                  28  DRVSEL   EQU  $C08A     ; $8A: Drive 1 / $8B: Drive 2
                  29  STROBE   EQU  $C08C     ; DATA IN bei READ
                  30  DATA     EQU  $C08D     ; DATA OUT bei WRITE
                  31  SREAD    EQU  $C08E     ; setzt "READ"
                  32  SWRITE   EQU  $C08F     ; setzt "WRITE"
                  33  *
                  34  * benutzte Speicherstellen in der Zeropage
                  35  *
                  36  ZVOLNO   EQU  $45       ; Timing in WRITE
                  37  SECTOR   EQU  $46
                  38  ZTRACK   EQU  $47
                  39  *
                  40          ORG  $2000
                  41  *
2000: 4C 0C 20    42          JMP  START     ; Einsprungvektoren für BASIC
2003: 4C 3D 20    43          JMP  DRIVEON
2006: 4C 68 20    44          JMP  DRIVEOFF
2009: 4C 0F 21    45          JMP  FORMAT
                  46  *
200C: A9 EF       47  START   LDA  #$EF      ; festgelegte Werte
200E: 8D 0D 21    48          STA  DELAY1    ; zum Hochzählen für die
2011: A9 D8       49          LDA  #$D8      ; Startzeit des Motors
2013: 8D 0E 21    50          STA  DELAY2
2016: 20 55 20    51          JSR  SELECT    ; Drive an
2019: 20 6F 20    52  ST1     JSR  RECAL     ; Arm zurück auf Track 00 und
201C: 20 4A 20    53          JSR  COUNTUP   ; "delay", bis Motor auf Touren
201F: 20 F3 21    54          JSR  RDADDR    ; lesen eines Address-Fields
2022: 90 06       55          BCC  NOTNEW    ; ging gut - keine neue Disk!
2024: A9 80       56          LDA  #$80
2026: 8D FF 02    57          STA  ERRNO     ; für BASIC: neue Disk
2029: 60          58          RTS            ; Drive läuft
                  59  *
202A: AD 09 03    60  NOTNEW  LDA  VOLFOUND
202D: 8D 02 03    61          STA  VOLNO
2030: A9 00       62          LDA  #0
2032: A9 00       63          LDA  #0
2034: CD 08 03    64          CMP  TRKFOUND  ; auf Track 00?
2037: D0 E0       65          BNE  ST1       ; darf eigentlich nicht passieren
2039: 8D FF 02    66          STA  ERRNO     ; für BASIC: bereits formatiert
203C: 60          67          RTS            ; Drive läuft
                  68  *
203D: A9 EF       69  DRIVEON LDA  #$EF      ; Motor-on-DELAY
203F: 8D 0D 21    70          STA  DELAY1
2042: A9 D8       71          LDA  #$D8
2044: 8D 0E 21    72          STA  DELAY2
2047: 20 55 20    73          JSR  SELECT    ; Drive on
                  74  *
204A: 2C 0E 21    75  COUNTUP BIT  DELAY2    ; bereits hochgezählt?
204D: 10 05       76          BPL  MOTORON
204F: 20 F5 20    77          JSR  ARMDELAY  ; zählt DELAY hoch
2052: F0 F6       78          BEQ  COUNTUP   ; "always"
2054: 60          79  MOTORON RTS
                  80  *
2055: AE 00 03    81  SELECT  LDX  SLOT16
2058: BD 89 C0    82          LDA  DRVON,X   ; Motor on
205B: BD 8E C0    83          LDA  SREAD,X   ; "READ MODE"
205E: 8A          84          TXA
205F: 18          85          CLC
2060: 6D 01 03    86          ADC  DRIVENO   ; 0 oder 1
2063: AA          87          TAX
2064: BD 8A C0    88          LDA  DRVSEL,X  ; Drive auswählen
2067: 60          89          RTS
                  90  *
2068: AE 00 03    91  DRIVEOFF LDX SLOT16
206B: BD 88 C0    92          LDA  DRVOFF,X
206E: 60          93          RTS
                  94  *
206F: AD 0B 03    95  RECAL   LDA  TRWANT    ; der gewünschte Track wird
2072: 48          96          PHA            ; gespeichert,
2073: A9 00       97          LDA  #00       ; "Track jetzt" auf $50 gesetzt
2075: 8D 0B 03    98          STA  TRWANT    ; und "Track gewünscht" auf 00
2078: A9 50       99          LDA  #$50      ; Folge: Der Arm wird auf
207A: 8D 0A 03    100         STA  TRNOW     ; Track 00 zurückgezogen
207D: 20 9E 20    101         JSR  MYSEEK
2080: A9 00       102         LDA  #00
2082: 8D 0A 03    103         STA  TRNOW
2085: 68          104         PLA
2086: 8D 0B 03    105         STA  TRWANT    ; Restore
2089: 60          106         RTS
                  107 *
208A: AD 0A 03    108 STEPIN  LDA  TRNOW     ; 1 Track nach innen
208D: 18          109         CLC
208E: 69 01       110         ADC  #1
2090: 8D 0B 03    111         STA  TRWANT
2093: D0 09       112         BNE  MYSEEK    ; "always"
                  113 *
                  114 *
2095: AD 0A 03    115 STEPOUT LDA  TRNOW     ; 1 Track nach außen
2098: 38          116         SEC
2099: E9 01       117         SBC  #1
209B: 8D 0B 03    118         STA  TRWANT
                  119 *
209E: AD 0A 03    120 MYSEEK  LDA  TRNOW     ; bei Standard-Drives
20A1: 2C 05 03    121         BIT  HFLAG     ; müssen pro Track zwei
20A4: 30 01       122         BMI  MS1       ; Armbewegungen ausgeführt
20A6: 0A          123         ASL            ; werden, bei 80-Track-Drives
20A7: 8D 0B 21    124 MS1     STA  TRNOW1    ; nur eine
20AA: AD 0B 03    125         LDA  TRWANT    ; hier sollte der Arm
20AD: 8D 0A 03    126         STA  TRNOW     ; hinterher sein
20B0: 2C 05 03    127         BIT  HFLAG
20B3: 30 01       128         BMI  MS2
20B5: 0A          129         ASL
20B6: 8D 0A 21    130 MS2     STA  TRWANT1   ; Ziel-Track
                  131 *
20B9: CD 0B 21    132         CMP  TRNOW1
20BC: F0 36       133         BEQ  ARMSET    ; bereits auf dem Track
                  134 *
20BE: AD 0B 21    135 SEEKABS LDA  TRNOW1
20C1: 8D 0C 21    136         STA  TRNOW2
20C4: 38          137         SEC
```

```
20C5: ED 0A 21  138          SBC  TRWANT1
20C8: F0 2D     139          BEQ  DLY1      ; letztes DELAY zum "auspendeln"
20CA: B0 05     140          BCS  MOVEOUT
20CC: EE 0B 21  141  MOVEIN  INC  TRNOW1
20CF: 90 03     142          BCC  PHASEON   ; "always"
20D1: CE 0B 21  143  MOVEOUT DEC  TRNOW1
                144  *
20D4: 38        145  PHASEON SEC            ; Phase zum Ziel-Track
20D5: 20 E7 20  146          JSR  SETARM    ; hin anschalten
20D8: 20 F5 20  147          JSR  ARMDELAY
                148  *
20DB: 18        149          CLC            ; danach die alte
20DC: AD 0C 21  150          LDA  TRNOW2    ; Phase aus
20DF: 20 EA 20  151          JSR  SETARM2
20E2: 20 F5 20  152          JSR  ARMDELAY
20E5: F0 D7     153          BEQ  SEEKABS   ; "always"
                154  *
20E7: AD 0B 21  155  SETARM  LDA  TRNOW1    ; Einsprung: neue Phase on
20EA: 29 03     156  SETARM2 AND  #03       ; Einsprung: alte Phase off
20EC: 2A        157          ROL            ; Carry bestimmt, ob "on/off"
20ED: 0D 00 03  158          ORA  SLOT16
20F0: AA        159          TAX
20F1: BD 80 C0  160          LDA  ARMBASE,X ; adressiert Stepper-Motor
20F4: 60        161  ARMSET  RTS
                162  *
20F5: A9 30     163  ARMDELAY LDA  #$30     ; fester Wert
20F7: A2 11     164  DLY1    LDX  #$11
20F9: CA        165  DLY2    DEX
20FA: D0 FD     166          BNE  DLY2
20FC: EE 0D 21  167          INC  DELAY1    ; für Motor on: Hochzähler
20FF: D0 03     168          BNE  DLY3
2101: EE 0E 21  169          INC  DELAY2
2104: 38        170  DLY3    SEC
2105: E9 01     171          SBC  #01
2107: D0 EE     172          BNE  DLY1
2109: 60        173          RTS
                174  *
                175  TRWANT1 DS   1
                176  TRNOW1  DS   1          ; Scratches für SEEKABS
                177  TRNOW2  DS   1
                178  *
                179  DELAY1  DS   1
                180  DELAY2  DS   1          ; Motor-on-DELAY
                181  *
                182  FORMAT  EQU  *
210F: AD 03 03  183          LDA  STRACK
2112: 8D 0B 03  184          STA  TRWANT
2115: 20 9E 20  185          JSR  MYSEEK    ; setzt Arm auf Start-Track
2118: A9 17     186  WRLOOP  LDA  #$17      ; inverses "W"
211A: 20 6B 21  187          JSR  PRTRACK
211D: 20 7C 21  188          JSR  GETESC    ; Test auf Abbruch
2120: 20 BB 22  189          JSR  WRTRACK
2123: A9 01     190          LDA  #1        ; "WRITE PROTECTED"?
2125: B0 3E     191          BCS  VFEXIT    ; mit ERR#1
2127: A9 AB     192          LDA  #"+"      ; Anzeige: WRITE beendet
2129: 20 6B 21  193          JSR  PRTRACK
212C: AD 0A 03  194          LDA  TRNOW
212F: CD 04 03  195          CMP  ETRACK    ; letzter Track fertig?
2132: F0 05     196          BEQ  VERIFY
2134: 20 8A 20  197          JSR  STEPIN    ; nächster Track
2137: F0 DF     198          BEQ  WRLOOP    ; "always"
                199  *
2139: A9 16     200  VERIFY  LDA  #$16      ; inverses "V"
213B: 20 6B 21  201          JSR  PRTRACK
213E: 20 93 21  202          JSR  VFTRACK
2141: B0 10     203          BCS  VFERR
2143: 20 7C 21  204          JSR  GETESC    ; Test auf Abbruch
2146: AD 0A 03  205          LDA  TRNOW
2149: CD 03 03  206          CMP  STRACK    ; letzter Track?
```

```
214C: F0 15     207          BEQ  VDONE     ; ok.
214E: 20 95 20  208          JSR  STEPOUT
2151: F0 E6     209          BEQ  VERIFY    ; "always"
2153: A9 03     210  VFERR   LDA  #3        ; Annahme: allgemeiner Fehler
2155: AE F2 21  211          LDX  SECTORS+15 ; Sektor $F ok?
2158: 10 0B     212          BPL  VFEXIT    ; nein
215A: AE E3 21  213          LDX  SECTORS+00 ; Sektor $0 ok?
215D: 30 06     214          BMI  VFEXIT
215F: A9 02     215          LDA  #2        ; Sektor $F ok,
2161: D0 02     216          BNE  VFEXIT    ; Sektor $0 nicht: ERR#2
                217  *
2163: A9 00     218  VDONE   LDA  #0
2165: 8D FF 02  219  VFEXIT  STA  ERRNO
2168: 4C 68 20  220          JMP  DRIVEOFF  ; RTS von da zu BASIC
                221  *
216B: 48        222  PRTRACK PHA            ; Character
216C: AD 0A 03  223          LDA  TRNOW
216F: C9 28     224  LIMIT   CMP  #$28
2171: 90 04     225          BCC  LM2       ; innerhalb der Zeile
2173: E9 28     226          SBC  #$28
2175: B0 F8     227          BCS  LIMIT     ; so lange, bis im Bereich 00-$27
2177: A8        228  LM2     TAY
2178: 68        229          PLA
2179: 91 28     230          STA  ($28),Y   ; "Print"
217B: 60        231          RTS
                232  *
217C: AD 00 C0  233  GETESC  LDA  $C000     ; Keyboard
217F: 10 11     234          BPL  NOTESC
2181: 8D 10 C0  235          STA  $C010     ; Keyboardstrobe löschen
2184: C9 9B     236          CMP  #$9B
2186: D0 0A     237          BNE  NOTESC
2188: 68        238          PLA            ; ansonsten wird die Return-
2189: 68        239          PLA            ; adresse heruntergeholt
218A: A9 05     240          LDA  #5        ; ERR#5: Abbruch
218C: 8D FF 02  241          STA  ERRNO
218F: 20 68 20  242          JSR  DRIVEOFF
2192: 60        243  NOTESC  RTS            ; bei <ESC>: zu BASIC
                244  *
                245  VFTRACK EQU  *
2193: A9 00     246          LDA  #0
2195: A2 0F     247          LDX  #$0F
2197: 8E E2 21  248          STX  SECSLEFT  ; "noch" 16 Sektoren übrig
219A: 9D E3 21  249  VFT1    STA  SECTORS,X
219D: CA        250          DEX            ; das Array "gefundene Sektoren"
219E: 10 FA     251          BPL  VFT1      ; wird gelöscht
                252  *
21A0: A9 21     253          LDA  #33       ; Zähler: jeder Sektor darf
21A2: 8D E1 21  254          STA  VFTRIES   ; maximal 2mal gelesen werden
                255  *
21A5: 20 F3 21  256  VFT2    JSR  RDADDR
21A8: 90 07     257          BCC  VFT4
21AA: CE E1 21  258  VFT3    DEC  VFTRIES
21AD: 10 F6     259          BPL  VFT2
21AF: 38        260          SEC
21B0: 60        261          RTS            ; aus
                262  *
21B1: AD 08 03  263  VFT4    LDA  TRKFOUND  ; richtiger Track?
21B4: CD 0A 03  264          CMP  TRNOW
21B7: F0 07     265          BEQ  VFT5
21B9: 68        266          PLA            ; Armposition falsch:
21BA: 68        267          PLA            ; zu hohe Tracknummer?
21BB: A9 04     268          LDA  #4        ; ERR#4: Positionsfehler
21BD: 4C 65 21  269          JMP  VFEXIT
                270  *
21C0: AE 07 03  271  VFT5    LDX  SECFOUND  ; gesetzt von RDADDR
21C3: BD E3 21  272          LDA  SECTORS,X
21C6: 30 E2     273          BMI  VFT3      ; schon gehabt
21C8: 20 5D 22  274          JSR  VFSECTOR  ; Daten verifizieren
21CB: B0 DD     275          BCS  VFT3
```

```
21CD: AE 07 03   276          LDX  SECFOUND
21D0: A9 80      277          LDA  #$80
21D2: 9D E3 21   278          STA  SECTORS,X
21D5: CE E2 21   279          DEC  SECSLEFT    ; noch Sektoren übrig?
21D8: 10 D0      280          BPL  VFT3
                 281  *
21DA: A9 AA      282          LDA  #"*"
21DC: 20 6B 21   283          JSR  PRTRACK
21DF: 18         284          CLC
21E0: 60         285          RTS
                 286  *
                 287  VFTRIES  DS   1
                 288  SECSLEFT DS   1
                 289  SECTORS  DS   16
                 290  *
                 291  RDADDR   EQU  *
21F3: A0 FC      292          LDY  #$FC
21F5: 8C 5A 22   293          STY  ADTRIES
21F8: AE 00 03   294          LDX  SLOT16      ; Slot-Nummer
                 295  *
21FB: C8         296  ADTRY1   INY
21FC: D0 05      297          BNE  ADLD1
21FE: EE 5A 22   298          INC  ADTRIES
2201: F0 55      299          BEQ  ADERR
                 300  *
2203: BD 8C C0   301  ADLD1    LDA  STROBE,X
2206: 10 FB      302          BPL  ADLD1
2208: C9 D5      303  ADTRY2   CMP  #$D5        ; $D5
220A: D0 EF      304          BNE  ADTRY1
220C: EA         305          NOP
220D: BD 8C C0   306  ADLD2    LDA  STROBE,X
2210: 10 FB      307          BPL  ADLD2
2212: C9 AA      308          CMP  #$AA        ; $AA
2214: D0 F2      309          BNE  ADTRY2
2216: A0 03      310          LDY  #03
2218: BD 8C C0   311  ADLD3    LDA  STROBE,X
221B: 10 FB      312          BPL  ADLD3
221D: C9 96      313          CMP  #$96        ; $96
221F: D0 E7      314          BNE  ADTRY2
2221: A9 00      315          LDA  #$00
2223: 8D 5B 22   316  ADDT0    STA  ADCHECK     ; CHECKSUM
2226: BD 8C C0   317  ADDT1    LDA  STROBE,X
2229: 10 FB      318          BPL  ADDT1
222B: 2A         319          ROL
222C: 8D 5C 22   320          STA  ADTEMP      ; erste Hälfte des Bytes
222F: BD 8C C0   321  ADDT2    LDA  STROBE,X    ; zweite Hälfte
2232: 10 FB      322          BPL  ADDT2
2234: 2D 5C 22   323          AND  ADTEMP
2237: 99 06 03   324          STA  ADDRBASE,Y  ; Vol-Nr./Track/Sector
223A: 4D 5B 22   325          EOR  ADCHECK
223D: 88         326          DEY
223E: 10 E3      327          BPL  ADDT0
2240: A8         328          TAY              ; zusammen mit CHECKSUM = 0?
2241: D0 15      329          BNE  ADERR
                 330  *
2243: BD 8C C0   331  ADTL1    LDA  STROBE,X
2246: 10 FB      332          BPL  ADTL1
2248: C9 DE      333          CMP  #$DE        ; $DE
224A: D0 0C      334          BNE  ADERR
224C: EA         335          NOP
224D: BD 8C C0   336  ADTL2    LDA  STROBE,X
2250: 10 FB      337          BPL  ADTL2
2252: C9 AA      338          CMP  #$AA        ; $AA
2254: D0 02      339          BNE  ADERR
                 340  *
2256: 18         341          CLC
2257: 60         342          RTS
                 343  *
2258: 38         344  ADERR    SEC
```

```
2259: 60         345          RTS
                 346  *
                 347  ADTRIES  DS   1
                 348  ADCHECK  DS   1
                 349  ADTEMP   DS   1
                 350  *
                 351  VFSECTOR EQU  *
225D: AE 00 03   352          LDX  SLOT16      ; Slot-Nummer
2260: A0 20      353          LDY  #$20        ; "Retries"
2262: 88         354  RDTRY1   DEY
2263: F0 54      355          BEQ  RDERR
2265: BD 8C C0   356  RDLD1    LDA  STROBE,X
2268: 10 FB      357          BPL  RDLD1
226A: C9 D5      358  RDTRY2   CMP  #$D5        ; $D5
226C: D0 F4      359          BNE  RDTRY1
226E: EA         360          NOP
226F: BD 8C C0   361  RDLD2    LDA  STROBE,X
2272: 10 FB      362          BPL  RDLD2
2274: C9 AA      363          CMP  #$AA        ; $AA
2276: D0 F2      364          BNE  RDTRY2
2278: A0 56      365          LDY  #$56
227A: BD 8C C0   366  RDLD3    LDA  STROBE,X
227D: 10 FB      367          BPL  RDLD3
227F: C9 AD      368          CMP  #$AD        ; $AD
2281: D0 E7      369          BNE  RDTRY2
                 370  *
2283: BD 8C C0   371  RDDT1    LDA  STROBE,X
2286: 10 FB      372          BPL  RDDT1
2288: C9 96      373          CMP  #$96        ; alle Sektoren sind leer, d.h.
228A: D0 2D      374          BNE  RDERR       ; enthalten nur $96
228C: 88         375          DEY
228D: D0 F4      376          BNE  RDDT1
228F: BD 8C C0   377  RDDT2    LDA  STROBE,X    ; zweiter Teil: "normale"
2292: 10 FB      378          BPL  RDDT2
2294: C9 96      379          CMP  #$96
2296: D0 21      380          BNE  RDERR
2298: C8         381          INY
2299: D0 F4      382          BNE  RDDT2
                 383  *
229B: BD 8C C0   384  RDCHECK  LDA  STROBE,X    ; die CHECKSUM ist
229E: 10 FB      385          BPL  RDCHECK     ; ebenfalls $96
22A0: C9 96      386          CMP  #$96
22A2: D0 15      387          BNE  RDERR
                 388  *
22A4: BD 8C C0   389  RDTL1    LDA  STROBE,X
22A7: 10 FB      390          BPL  RDTL1
22A9: C9 DE      391          CMP  #$DE        ; $DE
22AB: D0 0C      392          BNE  RDERR
22AD: EA         393          NOP
22AE: BD 8C C0   394  RDTL2    LDA  STROBE,X
22B1: 10 FB      395          BPL  RDTL2
22B3: C9 AA      396          CMP  #$AA        ; $AA
22B5: D0 02      397          BNE  RDERR
                 398  *
22B7: 18         399          CLC
22B8: 60         400          RTS
                 401  *
22B9: 38         402  RDERR    SEC
22BA: 60         403          RTS
                 404  *
                 405  *
                 406  WRTRACK  EQU  *
22BB: AE 00 03   407          LDX  SLOT16
22BE: BD 8D C0   408          LDA  DATA,X      ; Test auf "WRITE PROTECTED"
22C1: BD 8E C0   409          LDA  SREAD,X
22C4: 10 03      410          BPL  NOTWPROT
22C6: 38         411          SEC              ; "WRITE PROTECTED"!
22C7: B0 26      412          BCS  WREND       ; "always"
                 413  *
```

```
22C9: AD 02 03  NOTWPROT LDA  VOLNO
22CC: 85 45              STA  ZVOLNO     ; für Timing in WRITE
                 *
22CE: A9 FF              LDA  #$FF
22D0: 9D 8F C0           STA  SWRITE,X   ; ab jetzt: Zeit läuft!
22D3: 1D 8C C0           ORA  STROBE,X   ; erstes Sync-Byte
22D6: A9 00              LDA  #00
22D8: 85 46              STA  SECTOR
22DA: AD 0A 03           LDA  TRNOW
22DD: 85 47              STA  ZTRACK     ; Timing...
                 *
22DF: A9 FF              LDA  #$FF
22E1: A0 80              LDY  #$80       ; Syncs in GAP 1
                 *
22E3: 20 F6 22  NEXTSEC  JSR  WRSECTOR
22E6: A4 46              LDY  SECTOR
22E8: C0 10              CPY  #$10       ; letzter Sektor fertig?
22EA: A0 0E              LDY  #14        ; 16 Syncs zwischen den Sektoren
22EC: 90 F5              BCC  NEXTSEC
                 *
22EE: 18                 CLC
22EF: BD 8E C0  WREND    LDA  SREAD,X    ; zurück auf "READ MODE"
22F2: 1D 8C C0           ORA  STROBE,X
22F5: 60                 RTS
                 *
22F6: 24 00     WRSECTOR BIT  $00        ; Aufruf mit Acc = $FF
22F8: 20 D0 23           JSR  WRBYTE3
22FB: 24 00              BIT  $00
22FD: 2C 00 03           BIT  $300
                 *
2300: 24 00     GAP13    BIT  $00        ; GAP 1 und GAP 3: Y-Reg.
2302: 20 CD 23           JSR  WRBYTE1    ; bestimmt die Anzahl der
2305: 88                 DEY             ; Syncs
2306: F0 02              BEQ  ADDRLD
2308: D0 F6              BNE  GAP13
                 *
230A: 2C 00 03  ADDRLD   BIT  $300       ; 4 µsec
230D: A9 D5              LDA  #$D5       ; $D5
230F: 20 D0 23           JSR  WRBYTE3
2312: A9 AA              LDA  #$AA       ; $AA
2314: 20 CD 23           JSR  WRBYTE1
2317: A9 96              LDA  #$96       ; $96
2319: 20 CD 23           JSR  WRBYTE1
                 *
231C: A5 45              LDA  ZVOLNO     ; Volume-Nummer
231E: 20 BB 23           JSR  WRDB
2321: A5 47              LDA  ZTRACK     ; Track-Nummer
2323: 20 BB 23           JSR  WRDB
2326: A5 46              LDA  SECTOR     ; Sektor-Nummer
2328: 20 BB 23           JSR  WRDB
232B: AD 02 03           LDA  VOLNO
232E: 45 47              EOR  ZTRACK
2330: 45 46              EOR  SECTOR     ; und CHECKSUM, erster Teil
2332: 48                 PHA
2333: 4A                 LSR
2334: 09 AA              ORA  #$AA
2336: 9D 8D C0           STA  DATA,X
2339: DD 8C C0           CMP  STROBE,X
233C: 68                 PLA
233D: 09 AA              ORA  #$AA
233F: EA                 NOP
2340: 20 CD 23           JSR  WRBYTE1    ; CHECKSUM, zweiter Teil
                 *
2343: A9 DE              LDA  #$DE       ; $DE
2345: 20 CD 23           JSR  WRBYTE1
2348: A9 AA              LDA  #$AA       ; $AA
234A: 20 CD 23           JSR  WRBYTE1
234D: A9 EB              LDA  #$EB       ; $EB
234F: 20 CD 23           JSR  WRBYTE1
                 *
2352: A9 FF              LDA  #$FF       ; erstes Sync zwischen
```

```
2354: 20 CD 23           JSR  WRBYTE1    ; Address- und Data-Field
2357: A0 07              LDY  #7         ; + 7 weitere Syncs
2359: 24 00              BIT  $00
235B: EA                 NOP
235C: 24 00     GAP2     BIT  $00
235E: 20 CD 23           JSR  WRBYTE1
2361: 88                 DEY
2362: F0 02              BEQ  DATALD
2364: D0 F6              BNE  GAP2
                 *
2366: 2C 00 03  DATALD   BIT  $300       ; 4 µsec
2369: A9 D5              LDA  #$D5       ; $D5
236B: 20 D0 23           JSR  WRBYTE3
236E: A9 AA              LDA  #$AA       ; $AA
2370: 20 CD 23           JSR  WRBYTE1
2373: A9 AD              LDA  #$AD       ; $AD
2375: 20 CD 23           JSR  WRBYTE1
                 *
2378: A9 96              LDA  #$96       ; ab hier weicht das Verfahren
237A: 20 CD 23           JSR  WRBYTE1    ; stark von DOS 3.3 ab!
237D: A0 55              LDY  #$55
237F: 24 00              BIT  $00
2381: 2C 00 03  WRDAT1   BIT  $300       ; die ersten $55 Bytes
2384: EA                 NOP
2385: 20 D0 23           JSR  WRBYTE3    ; ("zusätzliche")
2388: 88                 DEY
2389: D0 F6              BNE  WRDAT1
238B: 24 00              BIT  $00
                 *
238D: 2C 00 03  WRDAT2   BIT  $300
2390: 20 D0 23           JSR  WRBYTE3    ; die nächsten $100 Bytes
2393: C8                 INY             ; ("normale")
2394: F0 02              BEQ  DATATL
2396: D0 F5              BNE  WRDAT2
                 *
2398: 2C 00 03  DATATL   BIT  $300
239B: A9 96              LDA  #$96       ; "CHECKSUM"
239D: 20 D0 23           JSR  WRBYTE3
                 *
23A0: A9 DE              LDA  #$DE       ; $DE
23A2: 20 CD 23           JSR  WRBYTE1
23A5: A9 AA              LDA  #$AA       ; $AA
23A7: 20 CD 23           JSR  WRBYTE1
23AA: A9 EB              LDA  #$EB       ; $EB
23AC: 20 CD 23           JSR  WRBYTE1
                 *
23AF: E6 46              INC  SECTOR
23B1: 2C 00 03           BIT  $300       ; insgesamt 9 µsec
23B4: 24 00              BIT  $00        ; JMP(!) folgt
                 *
23B6: A9 FF              LDA  #$FF       ; erstes Sync nach Data-Field,
23B8: 4C D0 23           JMP  WRBYTE3    ; für Zeit in Haupt-Loop
                 *
23BB: EA        WRDB     NOP
23BC: A8                 TAY             ; für die zweite Hälfte
23BD: 4A                 LSR
23BE: 09 AA              ORA  #$AA
23C0: 9D 8D C0           STA  DATA,X
23C3: DD 8C C0           CMP  STROBE,X
23C6: 98                 TYA
23C7: 09 AA              ORA  #$AA       ; zweite Hälfte
23C9: E6 46              INC  SECTOR     ; 5 µsec
23CB: C6 46              DEC  SECTOR     ; 5 µsec
                 *
23CD: 18        WRBYTE1  CLC             ; 2 µsec
23CE: 48        WRBYTE2  PHA             ; 3 µsec
23CF: 68                 PLA             ; 4 µsec
23D0: 9D 8D C0  WRBYTE3  STA  DATA,X     ; 5 µsec
23D3: DD 8C C0           CMP  STROBE,X   ; 4 µsec
23D6: 60                 RTS
983 Bytes
```

Für Ihre Unterlagen

Abonnement bestellt

am: _____

Vertrauensgarantie:
Ich habe davon Kenntnis genommen, daß ich die Bestellung schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1 innerhalb von 7 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels).

peeker
Leserservice
Postfach 10 28 69
6900 Heidelberg 1

Für Ihre Unterlagen

Folgende Bücher bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

peeker
Versandbuchhandlung
Postfach 10 28 69
6900 Heidelberg 1

Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____

_____

_____

_____

_____

am: _____

bei:

peeker
Softwareabteilung
Postfach 10 28 69
6900 Heidelberg 1

peeker
MAGAZIN FÜR APPLE-COMPUTER

Abo-Karte

Ja, ich möchte **peeker** abonnieren.
Liefern Sie mir **peeker** ab Ausgabe ..... (1985 erscheinen 11 Ausgaben – 1 Doppelnummer) zum Jahresbezugspreis von DM 72,– (Inland) incl. MwSt. Die Lieferung erfolgt frei Haus. Porto, Verpackung und Zustellgebühren übernimmt der Verlag. Der Jahresbezugspreis für das Ausland beträgt DM 72,– incl. MwSt., zzgl. DM 16,80 Versandspesen.

Ich wünsche jährliche Berechnung durch:

☐ Verlagsrechnung ☐ Abbuchung von meinem Bank- bzw. Postscheckkonto

Bank / PschA

Bankleitzahl Kto.-Nr.

Datum Unterschrift

peeker
MAGAZIN FÜR APPLE-COMPUTER

Buch-Shop

Bitte senden Sie mir gegen Rechnung folgende Bücher:

Menge	Autor, Titel	à DM	gesamt DM

Datum Unterschrift

peeker
MAGAZIN FÜR APPLE-COMPUTER

Software-Karte

Bitte senden Sie mir
gegen Rechnung folgende Apple-Programme:

☐ Peeker-Sammeldiskette, einzeln
Disk# _____, Disk# _____
Disk# _____, Disk# _____
Preis je Disk DM 28,– (einzeln)

☐ Peeker Sammeldiskette,
im Fortsetzungsbezug
ab Disk # _____
(Mindestbezug 6 Disketten)
Preis je Disk DM 20,–

☐ Apple DOS 3.3, Begleitdiskette, DM 28,–
☐ Apple ProDOS, Band 1, Begleitdiskette, DM 28,–
☐ Apple ProDOS, Band 2, Begleitdiskette, DM 28,–
☐ Apple Assembler, Begleitdiskette, DM 28,–
☐ ProDOS-Editor 1.0, Programm, DM 98,–
☐ MMU 2.0, Programm, DM 98,–
☐ INPUT 2.0, Programm, DM 98,–
☐ Softbreaker 1.0, Programm, DM 48,–
☐ DB-Meister, Programm, DM 290,–
☐ Superplot, Programm, DM 48,–
☐ Superquick, Programm, DM 48,–

Datum Unterschrift

Abo-Karte

Name

Firma

Abteilung

Straße

PLZ/Ort

Vertrauensgarantie:
Ich habe davon Kenntnis genommen, daß ich die Bestellung schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1 innerhalb von 7 Tagen widerrufen kann. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels).

Datum

Unterschrift

Verlagshinweis:
Das Abonnement verlängert sich zu den jeweils gültigen Bedingungen um ein Jahr, wenn es nicht 2 Monate vor Jahresende schriftlich gekündigt wird.

POSTKARTE

peeker
Leserservice

Postfach 10 28 69

6900 Heidelberg 1

Buch-Shop

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

peeker
Versandbuchhandlung

Postfach 10 28 69

6900 Heidelberg 1

Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

peeker
Softwareabteilung

Postfach 10 28 69

6900 Heidelberg 1

INPUT 2.0

Ein Bildschirm-Maskengenerator für DOS 3.3 und ProDOS

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7785-1021-5

„Input 2.0" liegt wahlweise in der Bank 1 oder Bank 2 der Language Card und wird durch einen kurzen Driver in den unteren 48K aufgerufen.
Für jedes Feld der Bildschirmmaske lassen sich u. a. definieren: Feldlänge (bis zu 255 Zeichen) – Vtab – Htab – Datentyp (insgesamt 8 Typen) – Scrollflag (starre oder dynamische Maske) – Ctrlflag – Füllflag – Löschflag – Bildschirmflag (40- oder 80-Z-Darstellung). Innerhalb eines Eingabefeldes besteht jeder denkbare Redigierkomfort (Insert, Delete, Rubout, Restore usw.).
Gerätevoraussetzung: Apple IIe oder IIc; ferner Apple II+ im 40-Zeichenmodus

MMU 2.0

Memory Managements Utilities

für die Apple IIe 64K-Karte DOS 3.3 (und ProDOS)

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7787-1023-1

Insgesamt enthält die neue „MMU 2.0"-Diskette über 25 Programme, die neue Einsatzmöglichkeiten für die Extended 80 Column Card (erweiterte 80-Z-Karte = 64K-Karte für den Apple IIe) erschließen. Ein Teil der Programme laufen auch auf dem Apple II Plus, doch ist „MMU 2.0" primär für 64K-Karte-Besitzer gedacht.
Gerätevoraussetzung: Apple IIe mit 64K-Karte oder IIc

Softbreaker 1.0

Eine softwaremäßige Interrupt-Utility für die Apple IIe 64K-Karte

von U. Stiehl

1984, Diskette und Manual, DM 48,–
ISBN 3-7785-1022-3

Softbreaker ist ein Assemblerprogramm, mit dessen Hilfe Programme, die sich von der 64K-Karte (= Extended 80 Column Card für den Apple IIe) starten lassen, unterbrochen, gespeichert, geladen und exakt an der Stelle der Unterbrechung fortgeführt werden können. Dadurch ist es auch möglich, Sicherungskopien von sogenannten kopiergeschützten Programmen herzustellen.
Mit Softbreaker unterbrochene Programme werden komplett, d. h. die ganzen 64K einschließlich Language Card, in nur ca. 11 Sekunden auf einer formatierten Diskette gesichert.
Gerätevoraussetzung: Apple IIe mit 64K-Karte

Hüthig Software Service,
Postfach 10 28 69, D-6900 Heidelberg

Bit-Editor

Zeichensatz-EPROMs für die Videx-Karte

von Joachim Klamt

Haben Sie Ihren Monitor schon „80-Zeichen-getestet"? Was bei einer Darstellung mit 40 Zeichen pro Zeile noch ein Genuß für das Auge ist, kann bei 80 Zeichen bereits zur Qual werden. Ein durchschnittliches Fernsehgerät zeigt waagrechte Tropfen und senkrechte Fädchen, ein Gleichheitszeichen mit einem Minussymbol darunter entspricht dem großen E, die an den Ecken angeknabberten D erscheinen wie O, Kleinbuchstaben sind noch schlechter zu erkennen. Qualitativ mittelmäßige Monitore lassen immerhin im Normalmodus ein einigermaßen ermüdungsfreies Lesen zu, doch selbst gute bis sehr gute Monitore enttäuschen spätestens bei inverser 80-Zeichendarstellung. Senkrechte Linien erscheinen nur hauchdünn.

Wenn Sie Ihren Geldbeutel nicht mit dem Kauf eines teuren Spitzenmonitors belasten wollen, dann schlage ich Ihnen meine preiswerte Lösung vor. Ändern Sie den Zeichensatz der 80-Zeichenkarte. Schreibt man die senkrechten Linien überproportional dick auf den Bildschirm, so erscheinen diese auf Grund der Unzulänglichkeiten des Systems gerade so breit wie die nicht geänderten, waagrechten Linien. Das im folgenden beschriebene Programm **BITEDITOR** unterstützt Sie beim „Umstricken" Ihres Zeichensatzes.

Mein ursprünglicher Wunsch war es, das an meiner Arbeitsstelle viel benutzte Multiplan auf den Apple-II-Rechnern mit 80 Zeichen pro Zeile optisch so ansprechend laufen lassen zu können wie mit 40 Zeichen, nämlich mit der Möglichkeit der inversen Darstellung. Das Feld, in welches Sie bei Multiplan gerade schreiben, erscheint invers. Bei 80-Zeichenkarten ohne inversen Zeichensatz wird das bearbeitende Feld durch Größer/Kleinersymbole eingegrenzt, welche aber Raum der Nachbarfelder beanspruchen und dort ein Zeichen unterdrücken. Besonders bei einigermaßen gefüllten Tabellen müssen Sie das so gekennzeichnete Arbeitsfeld erst mühsam suchen, während das inverse Feld sofort ins Auge sticht. Nach der Modifikation ergibt sich durch die hardwaremäßige Umschaltung zwischen deutschem und amerikanischem Zeichensatz für Multiplan ein weiterer Vorteil. Texte können mit dem deutschen Zeichensatz geschrieben werden, Formeln wirken wesentlich übersichtlicher mit den entsprechenden Klammern, welche ja dieselben ASCII-Werte wie die deutschen Sonderzeichen besitzen.

Auch Freunde des Applewriter werden dies zu schätzen wissen. Es muß nicht mehr mühsam der Maschinencode geknackt und geändert werden, damit der richtige Zeichensatz angesprochen wird, sondern man wählt deutsch oder amerikanisch mittels Schalter. Für Pascal gilt ähnliches wie für Multiplan, Texte in deutsch, übersichtliche Syntax mit amerikanischem Zeichensatz. Sinngemäß trifft dies auch für weitere Programme zu. In CP/M-BASIC entsprechen nun die Befehle INVERSE und NORMAL ihrer tatsächlichen Bedeutung.

Grundidee

Im allgemeinen zieren drei EPROMs das 80-Zeichen-Interface. In einem steht das Programm für die Karte, die beiden anderen enthalten den deutschen bzw. den amerikanischen Zeichensatz. Karten mit inverser Darstellungsmöglichkeit opfern einen der beiden länderspezifischen Zeichensätze. Das Hin- und Herschalten zwischen den Zeichensätzen erfolgt softwaremäßig, d.h. durch einen Befehl (z. B. Ctrl-Z 2, Ctrl-Z 3). Neuere 80-Zeichenkarten verwenden ein Zeichensatz-EPROM mit größerem Speicher. Es enthält den normalen und den inversen Zeichensatz desselben Landes. Der Befehl schaltet hier innerhalb des EPROMs auf den entsprechenden Speicherbereich (normal oder invers). Meine Modifikation läßt sich prinzipiell bei allen Arten von Karten durchführen, wobei zwischenzeitlich eine Karte in ähnlicher Form am Markt erschienen ist.

Nach der Änderung werden die Zeichensatz-EPROMs 2716 (2K) durch die größeren 2732 (4K) ersetzt. In dem beim Einschalten aktivierten EPROM stehen der normale deutsche und amerikanische Zeichensatz, im anderen durch Befehl anwählbaren EPROM der inverse deutsche und amerikanische. Die jeweils höchstwertige Adreßleitung A11 wird über einen gemeinsamen Pull-up-Widerstand an Vcc gelegt. Ein einpoliger Schließer genügt, um A11 an Masse zu legen, wodurch die andere Hälfte der EPROMs angesteuert und gelesen wird. So wählt der Schalter einen der beiden Zeichensätze, die Software entscheidet über normale oder inverse Darstellung.
Die EPROMs 2716 und 2732 sind praktisch pinkompatibel. Lediglich die Leiterbahnen zum Pin 21 (A11) sollten bei beiden 2732 aufgetrennt werden, um A11, wie im **Bild 1** gezeigt, neu zu verdrahten. Weitere Hardware-Änderungen sind nicht notwendig.

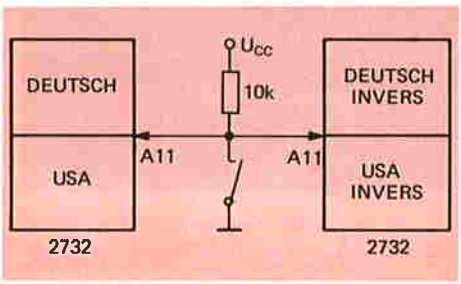


Bild 1

Das Prinzip der 80-Zeichendarstellung

Jedem am Bildschirm dargestellten Zeichen steht eine Matrix von 8 ∗ 9 Punkten zur Verfügung. Ein Zeichen besteht somit aus 72 einzelnen Punkten, die sich auf 9 Reihen mit 8 Spalten verteilen. Man kann sich jede Reihe als 8stellige Binärzahl vorstellen, wobei eine 0 einem nicht gesetzten Punkt entspricht und eine 1 einem gesetzten. Für jedes Zeichen wird ein Bereich von 16 8stelligen Binärzahlen reserviert, wobei nur die ersten 9 ausgewertet werden.
Das Betriebssystem der Karte (Programm-EPROM) veranlaßt das Auslesen der entsprechenden 9 Werte aus dem Zeichensatz-EPROM und deren punktweise Darstellung am Bildschirm.
Die neue softwaremäßige Umschaltung zwischen normaler und inverser Anzeige tritt an die Stelle der bisherigen länderspezifischen Zeichensatzumschaltung. Ein einmal invers geschriebener Text bleibt als solcher am Bildschirm erhalten, es sei denn, man überschreibt ihn im Normalmodus.

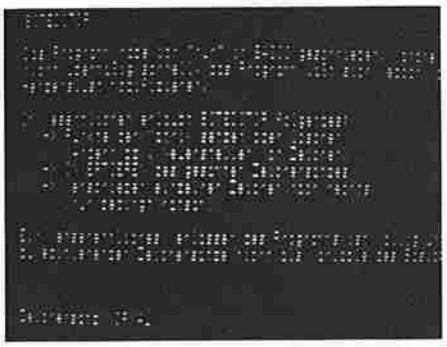


Foto A: Normaler Zeichensatz. Ein billiger Monitor in üblicher Einstellung zeigt bei 80-Zeichendarstellung dieses „bescheidene" Bild.

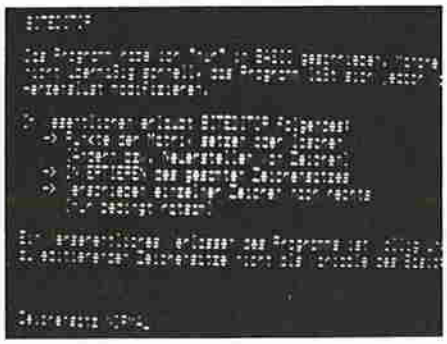


Foto B: Normaler Zeichensatz. Der Monitor ist für 80-Zeichendarstellung „optimal" eingestellt, jedoch dann nicht für 80 Zeichen invers.

Neue Zeichensatz-EPROMs

Wie ändere ich nun die Zeichensätze in den EPROMs? Falls Sie nicht gerade einen völlig neuen Zeichensatz kreieren wollen, so empfehle ich Ihnen folgende Methode. Lesen Sie die alten EPROMs, z.B. mit einem EPROM-Brenner, und speichern Sie die Inhalte als Binär-Files auf Diskette. Mit dem Programm BITEDITOR können Sie die Zeichensatz-Files nach Wunsch ändern, wieder auf Diskette schreiben und neue EPROMs schießen. Sie ersetzen Ihre alten Zeichensatz-EPROMs der 80-Zeichenkarte durch die neuen und können diese mit BITEDITOR testen (Menüpunkt TEST 80 Z). Wenn Sie den bis jetzt geänderten Zeichensatz-File laden, können Sie noch weitere Korrekturen vornehmen und wieder neu brennen. Verlieren Sie nicht den Mut, wenn Sie immer wieder einen Buchstaben entdekken, der Ihnen so noch nicht gefällt: ändern und neu brennen.
Ich habe nach wenigen Tagen den inversen Zeichensatz um eine Matrixlinie tiefer gesetzt. Die Unterlängen sind jetzt zwar kürzer, dafür kleben die inversen Zeichen nicht kontrastarm am oberen Zeilenrand. Sie werden oben und unten durch eine durchgehende Linie abgegrenzt, selbst dann, wenn ober- und unterhalb der Zeile keine inverse Darstellung erfolgt. Lediglich die Unterlängen reichen dann bis an den unteren Rand des Feldes. Die alphanumerischen Zeichenbereiche, die Sie tiefer setzen wollen, laden Sie einfach mit BLOAD um eine Adresse nach oben verschoben in den Speicher des Rechners. Später habe ich dann auch den normalen Zeichensatz tiefer gestellt, damit normale und inverse Darstellung innerhalb einer Zeile auf einer Linie liegen. Ein Verschieben des Bereiches mit den Grafiksymbolen erscheint nicht sinnvoll, da diese bis an die Ränder der Matrix reichen sollen. Nur so können bei der Aneinanderreihung durchgehende Figuren entstehen.
Ähnliche Überlegungen habe ich auch angestellt für ein seitliches Verschieben innerhalb der Matrix nach rechts. Links haben mehr Buchstaben einen senkrechten Strich als rechts. Der Kontrast zum nicht inversen Nachbarfeld bleibt öfter gewahrt, wenn die Buchstaben nicht linksbündig mit dem inversen Feld beginnen, sondern rechtsbündig mit diesem enden. So entstand das Unterprogramm „nach rechts verschieben", welches vom Menü aus mit VERSCHIEBEN aufgerufen wird. Das von BITEDITOR gerade angezeigte Zeichen wird um eine Matrixspalte nach rechts verschoben. Durch Einbinden in eine FOR-NEXT-Schleife könnten auch ganze Bereiche verschoben werden.
Der Versuch hat jedoch gezeigt, daß Zeichen nicht sauber geschrieben werden, wenn sie bis an den rechten Rand der Matrix reichen. Selbst der normale, nicht fette Zeichensatz leidet darunter. Tatsächlich werden nämlich nicht 8 ∗ 9, sondern nur 7 ∗ 9 Punkte der Matrix zur Zeichenbildung herangezogen. Die 8. Spalte kann zum Zeichenabstand gerechnet werden. Meine Änderungen beschränken sich deshalb auf ein Tieferstellen des normalen und inversen Zeichensatzes sowie auf das Verbreitern der senkrechten Zeichenanteile des inversen Zeichensatzes. Zeigt Ihr Bildschirmgerät bereits den normalen Zeichensatz schlecht an, so sollten Sie auch diesen fett brennen. Sie können ihn leicht durch Invertieren des inversen, fetten Zeichensatzes gewinnen. Durch Aufruf des Menüpunktes ↑INVERTIEREN (↑I = Ctrl-I) wird der gesamte Zeichensatz-File im Speicher invertiert.
Haben Sie Ihren deutschen Zeichensatz wunschgerecht erstellt, so laden Sie ihn hinter sich selbst in den Speicher (ab $5800) und ändern mit BITEDITOR die deutschen Sonderzeichen in die amerikanischen. Dabei sollte auch die Programmvariable FL$ = "800" (Zeile 4700) in FL$ = "1000" geändert werden, damit die hintereinander geladenen Files als ein einziger behandelt werden. Sie sollten nun zwei Files von je 18 Sektoren erstellt haben. Im ersten stehen der deutsche und amerikanische Zeichensatz in normaler Ausführung, im zweiten beide Zeichensätze invers.

Foto C: Fetter Zeichensatz. Darstellung am gleichen Monitor bei Verwendung des mit BITEDITOR verbesserten Zeichensatzes.

Foto D: Fett-inverser Zeichensatz. Die deutlichste Verbesserung zeigt sich bei der Darstellung des inversen Zeichensatzes im 80-Zeichenmodus. Der nicht fette Zeichensatz war am Foto praktisch unleserlich.

Die Bedienung des Programms

Das Applesoft-Programm BITEDITOR bietet im wesentlichen folgende Optionen:

- Punkte der Matrix setzen oder löschen (Ändern bzw. Neuerstellen von Zeichen),
- Invertieren des gesamten Zeichensatzes,
- Verschieben einzelner Zeichen nach rechts (nur bedingt ratsam).

Ein versehentliches Verlassen des Programms ist völlig unproblematisch, da die zu editierenden Zeichensätze nicht als Variablen des BASIC-Programms existieren, sondern als Binär-Files in den Rechner geladen werden. Auch das Ändern von Programmzeilen beeinflußt daher die Zeichensätze nicht.
Nach dem Starten des Programms erscheinen Hauptmenü und Arbeitsfeld (**Bild 2**) auf dem Bildschirm. Die Menüpunkte werden durch Eingabe des 1. Buchstabens aufgerufen:

LADEN – Sie werden nach dem Namen des zu ladenden Zeichensatz-Files gefragt. Der File steht danach im Speicher ab der Adresse $5000 (FS$ = "5000").
SPEICHERN – Sie können einen Namen eingeben, unter welchem Ihr geänderter Zeichensatz-File abgespeichert wird; <RETURN> allein übernimmt den beim Laden angegeben (VORGABE) Namen. Achten Sie auf die Größe der Variablen FL$, damit Sie den gesamten File sichern.
 # HEX-ADR – Durch die direkte Eingabe einer Hexadezimaladresse im Bereich von $5000 bis $6FFF werden die Speicherinhalte ab dieser Adresse angezeigt. Die Zeichen erscheinen immer richtig in der nachgebildeten Matrix, wenn die rechte Stelle der Hexadezimalzahl 0 ist. Die Länge des anzuzeigenden Bereichs legt die Variable LG (LG = 9) fest. Die Bildschirmaufteilung läßt Werte von 1 bis 16 zu.
Die Intervallvariable IV (IV = 16) gibt an, daß nach jeweils 16 Adressen der Bereich des nächsten Zeichens beginnt.
Pfeiltasten – Mit den Pfeiltasten kann vorwärts und rückwärts „geblättert" werden. Das geschieht durch Erhöhen oder Erniedrigen des Wertes für die momentan gültige Adresse um den Betrag IV. Der Inhalt des neuen Bereiches wird im Bitmuster-Fenster angezeigt.
LEERTASTE – Mit der Leertaste invertieren Sie das Feld unter dem Cursor und können somit Zeichen punktweise ändern.
I J K M – Das Bitmuster liegt innerhalb des Rahmens aus Sternchen. Das Pluszeichen ersetzt den blinkenden Cursor, welcher ja nicht erkennen ließe, ob an seinem Platz ein normales oder ein inverses Feld steht. I, J, K und M führen den Cursor wie im ESC-Modus in Applesoft.
NEUSTART – Gefallen Ihnen Ihre Änderungen nicht, so wird – sofern noch nicht <RETURN> gedrückt wurde – mit NEUSTART das unveränderte Zeichen aus dem Speicher geholt und zur Anzeige gebracht.
<RETURN> – Das geänderte Zeichen wird mit <RETURN> wie am Bildschirm sichtbar in den Rechnerspeicher übernommen.

WERTEAUSGABE – Bei Wahl des Menüpunktes WERTEAUSGABE erscheinen in der ersten Spalte des Arbeitsfeldes ADRESSE alle Adressen, sowie in der zweiten Spalte INHALT deren Werte. Die Umwandlungsroutinen am Anfang des Programms (HEX–DEZ und DEZ–BIN) lassen sich leicht in andere BASIC-Programme einbinden.

VERSCHIEBEN – Das angezeigte Zeichen wird um eine Matrixspalte nach rechts verschoben. Die Informationen der Spalte rechts außen erscheinen in der ersten Spalte links.

↑ INVERTIEREN – Bei Eingabe von Ctrl-I wird, wie schon erwähnt, der gesamte Zeichensatz invertiert. Ein inverser Zeichensatz wird dabei in einen normalen verwandelt. Der Bereich beginnt bei FS$ und endet bei (FS$ + FL$).

TEST 80 Z – Hierbei wird die 80-Zeichenkarte eingeschaltet. Sie können jetzt Zeichen per Tastendruck eingeben. Steckt ein bereits mit BITEDITOR geändertes EPROM in der installierten 80-Zeichenkarte, so können Sie Ihre Änderungen auswerten und nach dem Zurückschalten ins Hauptmenü eventuelle Korrekturen vornehmen. Sie ändern dabei natürlich nur den Zeichensatz-File im Rechner. Die nächste Auswertung kann erst nach erneutem Brennen des EPROMs erfolgen.
Die Kopfzeile zeigt drei mögliche Befehle an. Ctrl-I und Ctrl-N schalten auf inverse

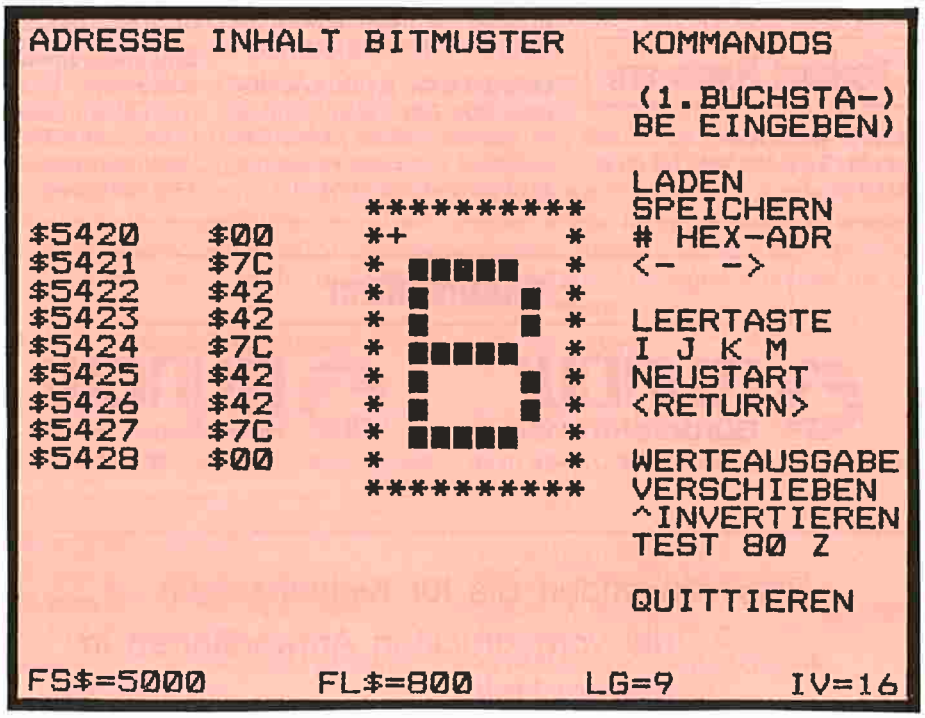


Bild 2

Lesen Sie weiter auf Seite 50

peeker Börse

Verkauf Software

Pascal-Toolbox und Anwendersoftware billig abzugeben. Info gegen Rückporto von Ernst Heinz Püttkampsweg 13, 2 Hamburg 52

Intelligenz-Test (IQ) für +/e/c, Disk mit ca. 170K Daten für 30 DM per Nachnahme bei: Frank Knab 6500 Mainz 43 Oppenheimer Str. 37

APPLE II: 'GIANT WORLD' ist da!! * Das neu Top-Adventure mit * * Spitzengrafik für riesigen * * Spielspaß! Es lohnt sich! * Für nur 89,– DM Vorkasse o.NN bei *** FANTASTIC-Software *** ** Grasweg 7, 2857 Langen 3 **

AWII-Manual (dtsch) + WPL DM 25 Privat-Dater (dBASE) DM 80, Biorhythmus (prof) DM 120 bei: Geßner, Haus 70, 8431 Kottingwörth Tel: 0 84 61/84 53

Über 100 HGR-Farben, HGR-Schrift in bel. Größe DM 40,– 02 61/6 35 86

** **Apple II Steuer** ** **ersetzt** alle Lohn- u. Eink. Steuertabellen + unbegrenzte Beträge und Kinder + Länderanpassung Kirchensteuer DM 120,– f. 1985 mit Nachlieferung 1986; Info kostenlos; W. Hagenmaier * Malerwinkel 7 8087 Türkenfeld *

Verkauf Hardware

256 K RAM Karte AP 17 IBS 599 DM ESW 103 VHB Tel: 05 21/ 87 04 24

Apple IIc, kompl. incl. ausführl. Literatur zu verk. DM 2500,– Tel: 0 21 61/4 14 60.

Fernschreiberinterface am Gameport m. Programm DM 79,– P. Benner, Hubertusstr. 131, 4150 Krefeld

MACINTOSH 128KB 2 Mon. alt VHB DM 6100,. Tel: 0 61 21/ 46 13 71

EPROM'S 2716 á 8 DM RAM's 4116-2 á 3 DM Tel: 06251/69742.

Verkaufe 128K Mac, Imagewriter, 2 Lfw. + umfangreiche Software (Originale) auch getrennt. v. Wennstein, Postf. 2057 Wentorf

TEAC 55F 560,– 55B 515,– * SSDD 10 St 42,– 96 tpi DS 78,– Lochverst. europ. Markenw. * Box Rglas. Schl. für 40–42,– für 80–49,– * Monitore 18Mhz 12" entsp. gr. ab 298,–. **VICO,** Selchower Str. 31, 1000 Bln 44

**** **Apple Supermodem** **** V21, V23 und Bell, 300–1200 Baud komplett auf einer Karte, mit Software startfertig nur 398 DM. Rolf Kiupel, Tel. 04 31/55 54 27

Apple II+ und Zubehör: Apple II+ 48K 900,–, 16KRam 80,–, Original Pal-Karte 190,–, Pal-Karte 90,– Accelerator II 790,–, alle Preise VHB. Tel: 0 62 21/7 48 32 oder 07 21/37 39 14.

256K-RAM Karte für Apple II/BASIS mit Treibersoftware DM 700,– W. Porten, 02 21/35 17 51.

MACINTOSH 512K-Erweiterung 890 DM 6 Mon. Garantie; 24 Stunden Service Händleranfr. erwünscht P. Clotten Remystr. 5; 5413 Bendorf 0 26 22/54 58

MACINTOSH 512K-Erweiterung 6 Mon. Garantie; 24 Stunden Service Händleranfr. erwünscht P. Clotten Remystr. 5; 5413 Bendorf 0 62 22/54 58

Verk. Apple II+ mit 2 Disk II, Contr., Monitor, Softw., VP 2000 Fr. Tel. 0 56/96 35 75 Schweiz b. Baden.

RAMKARTE (BASIS) 108/ APPLE) 256KB für 650,– DM – 1 EHRING-CONTROLLEER für 2×80TR. 130,– Tel: 02 01/ 50 21 92

Verkaufe Typenraddrucker Silverreed EXP 500 DIN A4 quer wenig gebraucht. Evtl. mit para. Int. VB 1100 DM, Tel: 0 21 22/5 55 29

Kontakte

Kennen Sie den Macintosh? Können Sie ihn programmieren und haben Sie interessante Ideen, dann sind Sie der richtige Partner für uns. Wir vermarkten Ihre Programme bundesweit – oder erteilen Programmieraufträge. Chiffre P1002

Verschiedenes

Anwendersoftware für Handwerk Tischlerei, Fensterbau, 7 Mitarb. gesucht. DOS-PRO-DOS- oder CPM Angebote an Heinz Heubrock Osterbauer 101, 4715 Ascheberg

Floppysubsystem für Apple II
2 × 3,5" TEAC FD-35F
max. 2 × 1MB
Autopatch-Controller
inkl. Manual und SW
ALU-Profil-Gehäuse m. Netzt.
komplett anschlußfertig
nur DM 2398,–

Telekommunikation
Anschlußkabel
Dataphon S21d
Terminalsoftware
komplett anschlußfertig
nur DM 398,–

TISCH & ZETTL GdbR
Elektronicvertrieb
Rosenstr. 33, 8034 Germering
Telefon 089/841 68 17

Suche amerikanische Aktienkurse auf Datenträger. Zeitraum 1978 – heute. Angebote an Tel. 0 72 51/ 10 30 68, H. Britting.

Ihr apple Partner in der Schweiz C&L Computershop Zentralstraße 93 5430 Wettingen Verlangen Sie unseren ungewöhnlichen Versandkatalog.

Progr. Ihre Eproms; 2508-2564 u. 27 16–2 71 28; 0 89/68 53 36

MC 3470 (der Leseverstärker auf dem Analogboard): DM 10,–, A. Deckers, PF 967, 7 Stuttgart 1

MAC mit Programmen sucht MAC ohne Programme! E. Schreiber 2070 Ahrensburg

Einkaufsführer

RUNOW Büroelektronik
Keithstr. 26 · 1 Berlin 30 · ☎ 0 30-26 111 26

RUNOW Büroelektronik GmbH
Bachstr. 104 · 2 HH 76 · ☎ 0 40-220 11 55

Bitte verwenden Sie für Kleinanzeigen die vorgedruckten Antwortkarten in diesem Heft.

Preisliste kostenlos!
Katalog DM 2,–

Unser Angebot im Juli

Chinon Laufwerk
(Test peeker 5/85)
DM 398,–

Profimax III
(= Lazar Ilze, Test peeker 6/85)
DM 1248,–

D.O.S. Computersysteme
Am Kühnbach 42, 7170 Schwäbisch Hall 11
Telefon (07 91) 5 17 36

Ulrich Stiehl

6502 leicht gemacht

Teil 1

1 Warum dieser Beitrag geschrieben wurde

Ziel dieses Beitrags ist es, den Assembler-Neuling mit der Programmierung des 6502-Prozessors vertraut zu machen. Dabei stand nicht die Vollständigkeit, sondern eine ballastfreie Gliederung im Vordergrund. Der in der Apple II Familie enthaltene 6502 eignet sich aus verschiedenen Gründen für die ersten Gehversuche in der Assemblerprogrammierung:

– Der Befehlssatz dieses Prozessors ist vergleichsweise einfach zu erlernen.
– Dieser Prozessor ist weitverbreitet und findet nicht nur im Apple, sondern z.B. auch in vielen Commodore Rechnern Anwendung.

Lehrbücher über Maschinensprache sind leider in der Regel völlig undidaktisch aufgebaut und beginnen meist mit der Binärmathematik, die für den Anfänger naturgemäß zunächst ein Buch mit sieben Siegeln ist. Deshalb sind viele Assembler-Anfänger frustriert und meinen, daß Maschinensprache nur etwas für „Eingeweihte" sei. In Wirklichkeit ist Assembler nicht wesentlich schwieriger als eine Hochsprache wie BASIC. Während jedoch z. B. BASIC sofort „Erfolgserlebnisse" vermittelt, dauert es in Assembler relativ lange, bis „sinnvolle" Programme geschrieben werden können. Dies rührt daher, daß aus Sicht der Maschinensprache BASIC-Befehle stets Makrobefehle sind, d. h. ein BASIC-Befehl wie HOME (= lösche den Bildschirm) setzt sich in Wirklichkeit aus zahlreichen elementaren Maschinenbefehlen zusammen, die isoliert betrachtet keine sinnvollen Aufgaben zu erfüllen scheinen. Maschinensprachlich gesehen besteht die Aufgabe des HOME-Befehls darin, in jeder einzelnen Zelle des Bildschirmspeichers eine Leertaste zu speichern und anschließend den Cursor in Zeile 1, Spalte 1 zu positionieren. Beim Apple ist der Bildschirmspeicher kein homogener Block, so daß komplizierte Algorithmen angewandt werden müssen, um versehentliches Löschen von Speicherbereichen, die nicht zum Bildschirm gehören, zu vermeiden. So wird verständlich, daß die Assembler-Programmierung des HOME-Befehls bereits ein größeres Unterfangen darstellt.

2 Wann ist das Programmieren in Assembler vorzuziehen?

Assembler-Programme sind fast immer erheblich kompakter und schneller als die entsprechenden Programme in irgendeiner höheren Programmiersprache. Dies liegt in der Tatsache begründet, daß ein Computer grundsätzlich nur Maschinenbefehle versteht. Bei einer Interpreter-Sprache wie BASIC muß der Interpreter jeden Befehl des BASIC-Quellenprogramms zunächst in eine Maschinenbefehlssequenz transformieren und diese dann ausführen. Bei einer Compiler-Sprache wie PASCAL — übrigens lassen sich auch BASIC-Programme kompilieren — erfolgt die Umwandlung in Maschinenbefehlssequenzen durch die Kompilation. Da dies jedoch ein mechanischer „Schema F"-Prozeß ist, entsteht ein aufgeblähtes, umständliches Maschinenprogramm, das wesentlich mehr Speicherraum einnimmt und erheblich langsamer abläuft als das entsprechende reine Assembler-Programm.

Grundsätzlich empfiehlt sich das Programmieren in Assembler stets dann, wenn maximale Kompaktheit und/oder maximale Geschwindigkeit des Programms erforderlich sind. Ferner ist Assembler angebracht, wenn die Hochsprache(n) nicht über diejenigen Befehle verfügen, die zur optimalen Lösung einer Programmieraufgabe erforderlich sind.

Die beiden Applesoft-BASIC-Beispiele in Tabelle 1 veranschaulichen den Zeitgewinn bei Verwendung der Assembler-Sprache. Bei Programm 1 handelt es sich um ein Zählprogramm mit Integer-Zahlen. Da z.B. der TASC-Compiler **(ein Compiler für den Apple II Plus)** über echte Integer-Arithmetik verfügt, ist das entsprechende Assembler-Programm „nur" neunmal schneller als die TASC-Version.

Tabelle 1: Geschwindigkeitsvergleich (Dauer in Sekunden)

Programm	Assembler	TASC-Compiler	Hayden-Compiler	Applesoft
1	.5	4.5	15.5	38
2	.25	13	23	64

```
10 REM PROGRAMM 1
20 S% = 1 : E% = 10000
30 FOR C = S% TO E% : REM
   START BIS ENDE
40 A% = C + E% : REM ADDIERE
   ZÄHLER ZU E%
50 NEXT
```

```
10 REM PROGRAMM 2
20 S% = 8192 : E% = 24575 :
   Z% = 0
30 FOR C = S% TO E%
40 POKE C,Z% : REM LÖSCHE
   SPEICHERSTELLE
50 NEXT
```

ASCII-Tabelle

NUL	@ §	00	00000000	000	@ §	40	01000000	064	@ §	80	10000000	128	@ §	C0	11000000	192
	A	01	00000001	001	A	41	01000001	065	A	81	10000001	129	A	C1	11000001	193
	B	02	00000010	002	B	42	01000010	066	B	82	10000010	130	B	C2	11000010	194
	C	03	00000011	003	C	43	01000011	067	C	83	10000011	131	C	C3	11000011	195
EOT	D	04	00000100	004	D	44	01000100	068	D	84	10000100	132	D	C4	11000100	196
	E	05	00000101	005	E	45	01000101	069	E	85	10000101	133	E	C5	11000101	197
	F	06	00000110	006	F	46	01000110	070	F	86	10000110	134	F	C6	11000110	198
BELL	G	07	00000111	007	G	47	01000111	071	G	87	10000111	135	G	C7	11000111	199
←	H	08	00001000	008	H	48	01001000	072	H	88	10001000	136	H	C8	11001000	200
TAB	I	09	00001001	009	I	49	01001001	073	I	89	10001001	137	I	C9	11001001	201
↓	J	0A	00001010	010	J	4A	01001010	074	J	8A	10001010	138	J	CA	11001010	202
↑	K	0B	00001011	011	K	4B	01001011	075	K	8B	10001011	139	K	CB	11001011	203
FF	L	0C	00001100	012	L	4C	01001100	076	L	8C	10001100	140	L	CC	11001100	204
RTN	M	0D	00001101	013	M	4D	01001101	077	M	8D	10001101	141	M	CD	11001101	205
SO	N	0E	00001110	014	N	4E	01001110	078	N	8E	10001110	142	N	CE	11001110	206
SI	O	0F	00001111	015	O	4F	01001111	079	O	8F	10001111	143	O	CF	11001111	207
	P	10	00010000	016	P	50	01010000	080	P	90	10010000	144	P	D0	11010000	208
XON	Q	11	00010001	017	Q	51	01010001	081	Q	91	10010001	145	Q	D1	11010001	209
	R	12	00010010	018	R	52	01010010	082	R	92	10010010	146	R	D2	11010010	210
XOFF	S	13	00010011	019	S	53	01010011	083	S	93	10010011	147	S	D3	11010011	211
	T	14	00010100	020	T	54	01010100	084	T	94	10010100	148	T	D4	11010100	212
→	U	15	00010101	021	U	55	01010101	085	U	95	10010101	149	U	D5	11010101	213
	V	16	00010110	022	V	56	01010110	086	V	96	10010110	150	V	D6	11010110	214
	W	17	00010111	023	W	57	01010111	087	W	97	10010111	151	W	D7	11010111	215
	X	18	00011000	024	X	58	01011000	088	X	98	10011000	152	X	D8	11011000	216
	Y	19	00011001	025	Y	59	01011001	089	Y	99	10011001	153	Y	D9	11011001	217
	Z	1A	00011010	026	Z	5A	01011010	090	Z	9A	10011010	154	Z	DA	11011010	218
ESC	[Ä	1B	00011011	027	[Ä	5B	01011011	091	[Ä	9B	10011011	155	[Ä	DB	11011011	219
	\ Ö	1C	00011100	028	\ Ö	5C	01011100	092	\ Ö	9C	10011100	156	\ Ö	DC	11011100	220
	] Ü	1D	00011101	029	] Ü	5D	01011101	093	] Ü	9D	10011101	157	] Ü	DD	11011101	221
	↑	1E	00011110	030	↑	5E	01011110	094	↑	9E	10011110	158	↑	DE	11011110	222
	_	1F	00011111	031	_	5F	01011111	095	_	9F	10011111	159	_	DF	11011111	223
		20	00100000	032	'	60	01100000	096		A0	10100000	160	'	E0	11100000	224
	!	21	00100001	033	a	61	01100001	097	!	A1	10100001	161	a	E1	11100001	225
	"	22	00100010	034	b	62	01100010	098	"	A2	10100010	162	b	E2	11100010	226
	#	23	00100011	035	c	63	01100011	099	#	A3	10100011	163	c	E3	11100011	227
	$	24	00100100	036	d	64	01100100	100	$	A4	10100100	164	d	E4	11100100	228
	%	25	00100101	037	e	65	01100101	101	%	A5	10100101	165	e	E5	11100101	229
	&	26	00100110	038	f	66	01100110	102	&	A6	10100110	166	f	E6	11100110	230
	'	27	00100111	039	g	67	01100111	103	'	A7	10100111	167	g	E7	11100111	231
	(	28	00101000	040	h	68	01101000	104	(	A8	10101000	168	h	E8	11101000	232
	)	29	00101001	041	i	69	01101001	105	)	A9	10101001	169	i	E9	11101001	233
	*	2A	00101010	042	j	6A	01101010	106	*	AA	10101010	170	j	EA	11101010	234
	+	2B	00101011	043	k	6B	01101011	107	+	AB	10101011	171	k	EB	11101011	235
	,	2C	00101100	044	l	6C	01101100	108	,	AC	10101100	172	l	EC	11101100	236
	–	2D	00101101	045	m	6D	01101101	109	–	AD	10101101	173	m	ED	11101101	237
	.	2E	00101110	046	n	6E	01101110	110	.	AE	10101110	174	n	EE	11101110	238
	/	2F	00101111	047	o	6F	01101111	111	/	AF	10101111	175	o	EF	11101111	239
	0	30	00110000	048	p	70	01110000	112	0	B0	10110000	176	p	F0	11110000	240
	1	31	00110001	049	q	71	01110001	113	1	B1	10110001	177	q	F1	11110001	241
	2	32	00110010	050	r	72	01110010	114	2	B2	10110010	178	r	F2	11110010	242
	3	33	00110011	051	s	73	01110011	115	3	B3	10110011	179	s	F3	11110011	243
	4	34	00110100	052	t	74	01110100	116	4	B4	10110100	180	t	F4	11110100	244
	5	35	00110101	053	u	75	01110101	117	5	B5	10110101	181	u	F5	11110101	245
	6	36	00110110	054	v	76	01110110	118	6	B6	10110110	182	v	F6	11110110	246
	7	37	00110111	055	w	77	01110111	119	7	B7	10110111	183	w	F7	11110111	247
	8	38	00111000	056	x	78	01111000	120	8	B8	10111000	184	x	F8	11111000	248
	9	39	00111001	057	y	79	01111001	121	9	B9	10111001	185	y	F9	11111001	249
	:	3A	00111010	058	z	7A	01111010	122	:	BA	10111010	186	z	FA	11111010	250
	;	3B	00111011	059	{ ä	7B	01111011	123	;	BB	10111011	187	{ ä	FB	11111011	251
	<	3C	00111100	060	| ö	7C	01111100	124	<	BC	10111100	188	| ö	FC	11111100	252
	=	3D	00111101	061	} ü	7D	01111101	125	=	BD	10111101	189	} ü	FD	11111101	253
	>	3E	00111110	062	~ ß	7E	01111110	126	>	BE	10111110	190	~ ß	FE	11111110	254
	?	3F	00111111	063	DEL	7F	01111111	127	?	BF	10111111	191	DEL	FF	11111111	255

Bei dem Programm 2 geht es um das Löschen eines Speicherbereiches (ähnlich wie bei Eingabe von HOME). Hierfür gibt es jedoch keinen entsprechenden BASIC-Befehl, so daß eine umständliche Simulation mit der POKE-Anweisung erforderlich ist. Infolgedessen ist die Assembler- über 50mal schneller als die TASC-Version.

Per Umkehrschluß läßt sich aus diesen beiden Beispielen folgern, daß Assembler-Programmierung stets dann überflüssig ist, wenn die Hochsprache exakt über diejenigen Makrobefehle verfügt, die maschinensprachlich ausgeführt werden sollen. So führt z. B. die Assembler-Programmierung des HOME-Befehls zu keiner Geschwindigkeitssteigerung, da dieser bereits als Maschinenroutine vorliegt, die der BASIC-Interpreter lediglich aufzurufen braucht.

3 Dezimale und hexadezimale Zahlen

Dezimale Zahlen haben die Basis 10 und setzen sich aus den Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 zusammen.

Die Dezimalzahl 123 bedeutet

$$10^{\overset{1}{2}} \cdot 1 + 10^{\overset{2}{1}} \cdot 2 + 10^{\overset{3}{0}} \cdot 3 = 123 \text{ dezimal}$$

Hexadezimale Zahlen haben die Basis 16 und setzen sich aus den Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F zusammen.

Die hexadezimale Zahl $123 bedeutet

$$16^{\overset{1}{2}} \cdot 1 + 16^{\overset{2}{1}} \cdot 2 + 16^{\overset{3}{0}} \cdot 3 = 291 \text{ dezimal}$$

Hexadezimalen Zahlen wird üblicherweise ein Dollar-Zeichen vorangestellt, um Verwechslungen mit Dezimalzahlen zu vermeiden. **Tabelle 2** gibt den Zusammenhang zwischen Hexadezimal- und Dezimalzahlen sowie die entsprechenden ASCII-Zeichen (American Standard Code for Information Interchange) für den Bereich 0 bis 255 wieder. Wenn Zahlen im Bereich von $100 (dezimal 256) bis $FFFF (dezimal 65535) umzuwandeln sind, hilft folgender Algorithmus weiter.

Beispiel:

$ABCD

Aufspaltung:	AB	+ CD
Addition:	171·256	+ 205·1
		= 43981 dezimal

(Das binäre Zahlensystem wird erst später eingeführt.)

4 Byte, Register, Adresse und Befehl

Byte

Ein Byte läßt sich definieren als eine hexadezimale Zahl im Bereich $00 bis $FF (dezimal 0 bis 255) und setzt sich aus 2 hexadezimalen Ziffern, Halbbytes oder Nibbles zusammen. Z. B. besteht das Byte $AB aus dem höherwertigen (= linken) Nibble $A und aus dem niederwertigen (= rechten) Nibble $B.

Register

Alles was in einem Mikrocomputer geschieht, geschieht durch den Mikroprozessor. Zur Vereinfachung nehmen wir zunächst an, daß der 6502 aus drei Registern oder Mikroprozessor-Speicherstellen besteht, die Bytes laden, speichern oder sonstwie manipulieren können. Diese drei Register heißen:

A oder Akkumulator
X oder Index Register X
Y oder Index Register Y.

Adresse

Der Speicher (RAM = Lese-Schreib-Speicher, ROM = Nur-Lese-Speicher), der vom 6502 adressiert werden kann, läßt sich anschaulich als eine Kette von $10000 oder 65536 Zellen (Speicherplätzen oder Speicherstellen) vorstellen, von denen jede einzelne 1 Byte enthalten kann. Diese Speicherstellen sind von $0000 bis $FFFF (0 bis 65535) numeriert. Die Adresse ist die Nummer einer Speicherstelle. Es ist wichtig, zwischen der Nummer einer Speicherstelle und ihrem Wert — bildlich zwischen der Hausnummer und dem Inhalt des Hauses — zu unterscheiden. Z. B. kann die Speicherstelle $00FF das Byte $AA und die Speicherstelle $00AA das Byte $FF enthalten.

Eine Adresse umfaßt beim 6502 grundsätzlich 2 Bytes. Beispielsweise besteht die Adresse $100A aus dem höherwertigen (= linken) Byte $10 und dem niederwertigen (= rechten) Byte $0A. Zusammenfassend gilt:

a) Eine Speicherstelle enthält 1 Byte ($00 — $FF)
b) Eine Adresse umfaßt 2 Bytes ($0000 — $FFFF).

(Als Sonderfall kann im Bereich $0000 — $00FF eine Adresse auch durch 1 Byte ausgedrückt werden. Weil hier das höherwertige Byte implizit null ist ($00), wird dieser Bereich als Zero-Page oder Nullseite bezeichnet.)

Befehl

Ein maschinensprachliches Programm ist eine Folge von Befehlen. Der Befehlssatz des 6502 Mikroprozessors umfaßt 56 verschiedene Anweisungen. Der Zweck des Prozessors besteht darin, diese zu dekodieren und auszuführen. Bei den meisten Befehlen wird der Inhalt einer Speicherstelle durch einen Registerinhalt verändert und umgekehrt.

5 Laden und Speichern

LDA = Load Accumulator = Lade den Akkumulator mit dem Inhalt, der sich in einer bestimmten Speicherstelle befindet, und

STA = Store Accumulator = Speichere den Inhalt des Akkumulators in eine bestimmte Speicherstelle

sind Beispiele von 6502 Befehlen. Die Instruktionen LDA und STA lassen sich wie folgt verdeutlichen:

LDA $035A lädt den Akkumulator mit dem Byte, das sich in der Speicherstelle $035A befindet (z.B. $FF). Nach dem Ladevorgang befindet sich im Akkumulator dann das Byte $FF; gleichzeitig bleibt $FF in der Speicherzelle $035A erhalten (Abbildung 1a).

STA $0361 überträgt den Inhalt des Akkumulators (jetzt $FF) in die Speicherstelle $0361; $FF bleibt gleichzeitig im Akkumulator erhalten (Abbildung 1b).

Lade- und Speicherbefehle verändern somit den Inhalt der Speicherzelle, aus der geladen wird, nicht. (Zum „Löschen" einer Speicherstelle muß in ihr $00 gespeichert werden.)

Die Befehlssequenz LDA $035A STA $0361 stellt das Maschinenprogramm dar und muß selbst irgendwo im Speicher abgelegt sein, z. B. ab Adresse $0300 (Abbildung 1c). Trifft der Mikroprozessor auf das Byte $AD in Speicherstelle $0300, dekodiert er $AD als LDA-Befehl und faßt die nachfolgenden 2 Bytes als Adresse auf, die beim 6502 stets „umgedreht" sind („low byte first"), also AD 5A 03 statt AD 03 5A.

Nach Ausführen dieses Befehls (sogenannter 3-Byte-Befehl: 1 Byte für Befehlscode + 2 Bytes für Adresse) gelangt der Mikroprozessor zur Speicherstelle $0303, in der sich $8D befindet, das als STA dekodiert wird. Die nachfolgenden zwei Bytes werden wiederum als Adresse interpretiert.

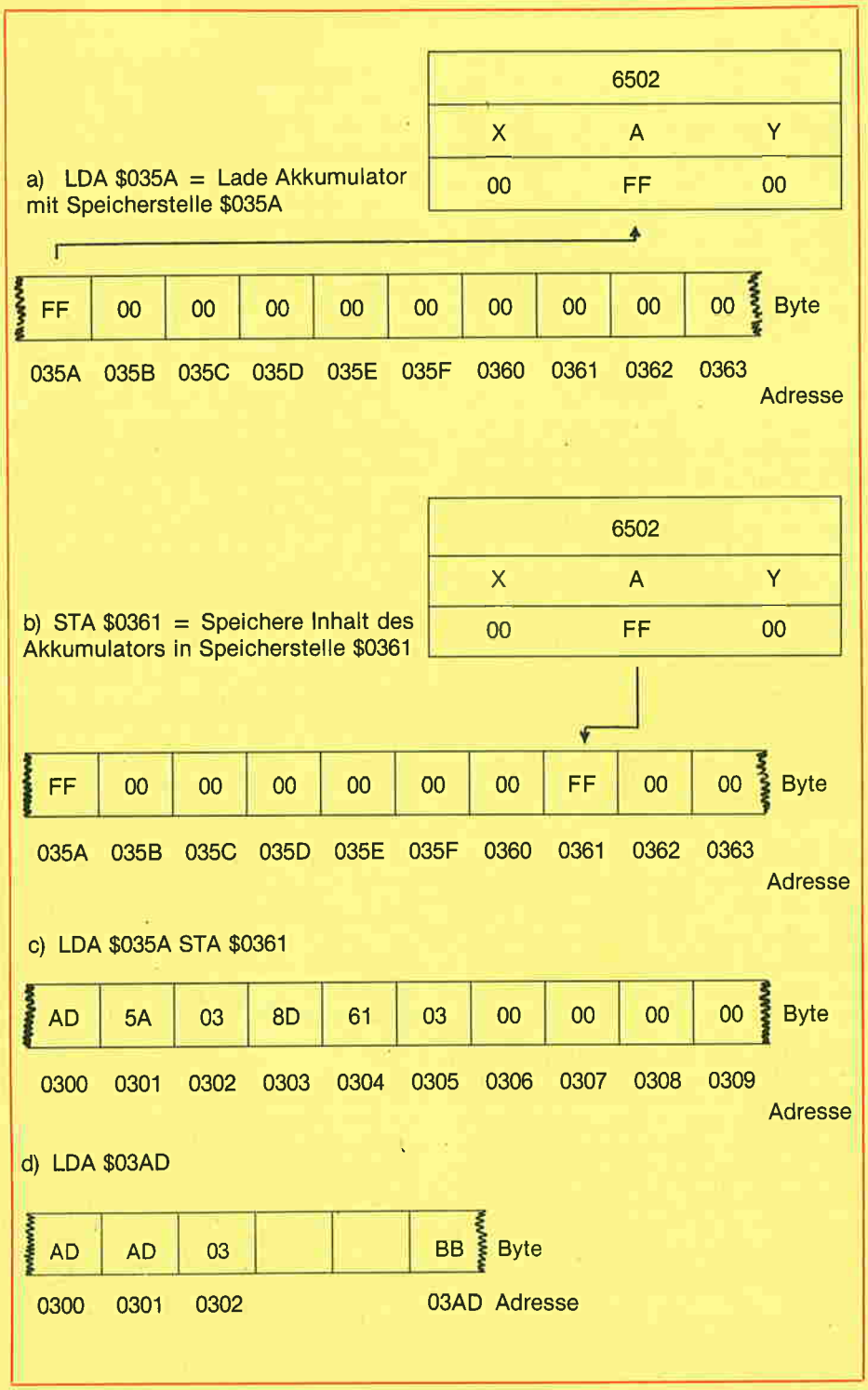


Abbildung 1: Darstellung der 6502 Befehle LDA und STA

Tabelle 3: Einteilung des Quell-Codes.

Feld 1	Feld 2	Feld 3	Feld 4
LABEL	OP-CODE	OPERAND	COMMENT
	ORG	768	;origin
ADDRESS1	EQU	$035A	;equates
ADDRESS2	EQU	$0361	;equates
START	LDA	ADDRESS1	;loading
	STA	ADDRESS2	;storing
END	RTS		

Trifft der Mikroprozessor z. B. auf die Byte-Folge AD AD 03 (Abbildung 1d), interpretiert er das erste $AD als LDA und das zweite $AD als Teil der 2-Byte-Adresse $03AD. Wenn sich in dieser Speicherstelle nun $BB befindet, so hängt es vom Sinn des Programms ab, ob BB als Hexadezimalzahl $BB, als Dezimalzahl 187, als ASCII-Zeichen „;" oder sonstwie interpretiert wird.

$AD für LDA ist ein sogenannter Operations-Code (Op-Code), d. h. eine hexadezimale Zahl, die vom Mikroprozessor als ein gültiger Befehl interpretiert wird. „LDA" selbst in ein sogenanntes Befehlskurzwort (Mnemonic). Die Befehlskurzwörter sind standardisiert und bestehen beim 6502 grundsätzlich aus 3 Buchstaben.

Teil 2

6 Assembler und Maschinensprache

Im vorangegangenen Teil wurde noch nicht genau zwischen Assembler(sprache) und Maschinensprache unterschieden. Ein Maschinenprogramm (= ein in Maschinensprache geschriebenes Programm) ist eine Folge von hexadezimalen Zahlen, z. B. „AD 5A 03 8D 61 03". Wir erinnern uns, daß dieses hexadezimale Kauderwelsch „LDA $035A STA $0361" bedeutet. Die hexadezimalen Werte eines Maschinenprogramms lassen sich zwar von BASIC aus „poken", etwa in der Form

```
POKE 768, 173 : REM $AD
POKE 769, 90  : REM $5A
POKE 770, 3   : REM $03 usw.,
```

doch ist diese Prozedur umständlich und damit für längere Maschinenprogramme nicht zu empfehlen.

Ein Assembler ist ein Hilfsprogramm, das aus zwei Modulen besteht, dem Editor und dem Assembler im eigentlichen Sinn. Der Editor dient zum Erstellen und Redigieren des Quell-Codes (z. B. „LDA $035A STA $0361"), der Assembler zur Umwandlung des Quell-Codes in den Objekt-Code (z. B. AD 5A 03 8D 61 03). Assembler sind meist zeilenorientiert (wie BASIC), wobei jede Befehlszeile des Quell-Codes in vier Felder eingeteilt ist **(Tabelle 3):**

Feld 1: „Labels" sind Marken oder Namen für Adressen, z. B. „ADDRESS1 EQU $035A" (kann entfallen).

Feld 2: „Mnemonics" sind leicht einprägbare Befehlskürzel für die Op-Codes, die beim 6502 regelmäßig aus

Tabelle 4: 6502 Mnemonics nach Funktionen mit den Op-Codes für die Adressierungsarten, der Anzahl der Takte, der Anzahl der Bytes sowie den beeinflußten Statusflags.

Function	Mnemon.	Accu.		Immedi.		Zero-P.		Zero,X		Zero,Y		Absol.		Abs,X		Abs,Y		(Ind,X)		(Ind),Y		Implied		Relative		Indirect		Status
		Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	Op	Cy	N Z C I D V
Load	LDA			A9	2	A5	3	B5	4			AD	4	BD	4*	B9	4*	A1	6	B1	5*							N Z – – – –
	LDX			A2	2	A6	3			B6	4	AE	4			BE	4*											N Z – – – –
	LDY			A0	2	A4	3	B4	4			AC	4	BC	4*													N Z – – – –
Store	STA					85	3	95	4			8D	4	9D	5	99	5	81	6	91	6							– – – – – –
	STX					86	3			96	4	8E	4															– – – – – –
	STY					84	3	94	4			8C	4															– – – – – –
Transfer	TAX																					AA	2					N Z – – – –
	TXA																					8A	2					N Z – – – –
	TAY																					A8	2					N Z – – – –
	TYA																					98	2					N Z – – – –
Stack Ptr	TSX																					BA	2					N Z – – – –
	TXS																					9A	2					– – – – – –
Stack	PHA																					48	3					– – – – – –
	PLA																					68	4					N Z – – – –
Status R.	PHP																					08	3					– – – – – –
	PLP																					28	4					N Z C I D V
Flags	CLC																					18	2					– – 0 – – –
	SEC																					38	2					– – 1 – – –
	CLI																					58	2					– – – 0 – –
	SEI																					78	2					– – – 1 – –
	CLD																					D8	2					– – – – 0 –
	SED																					F8	2					– – – – 1 –
	CLV																					B8	2					– – – – – 0
Jump	JMP											4C	3													6C	5	– – – – – –
	JSR											20	6															– – – – – –
Return	RTS																					60	6					– – – – – –
	RTI																					40	6					N Z C I D V
Compare	CMP			C9	2	C5	3	D5	4			CD	4	DD	4*	D9	4*	C1	6	D1	5*							N Z C – – –
	CPX			E0	2	E4	3					EC	4															N Z C – – –
	CPY			C0	2	C4	3					CC	4															N Z C – – –
	BIT					24	3					2C	4															– Z – – – –
Branch	BMI																							30	2+			– – – – – –
N = 0	BPL																							10	2+			– – – – – –
	BEQ																							F0	2+			– – – – – –
Z = 0	BNE																							D0	2+			– – – – – –
	BCS																							B0	2+			– – – – – –
C = 0	BCC																							90	2+			– – – – – –
	BVS																							70	2+			– – – – – –
V = 0	BVC																							50	2+			– – – – – –
Increment	INC					E6	5	F6	6			EE	6	FE	7													N Z – – – –
	INX																					EB	2					N Z – – – –
	INY																					C8	2					N Z – – – –
Decrement	DEC					C6	5	D6	6			CE	6	DE	7													N Z – – – –
	DEX																					CA	2					N Z – – – –
	DEY																					88	2					N Z – – – –
Add/Subt.	ADC			69	2	65	3	75	4			6D	4	7D	4*	79	4*	61	6	71	5*							N Z C – – V
	SBC			E9	2	E5	3	F5	4			ED	4	FD	4*	F9	4*	E1	6	F1	5*							N Z C – – V
Boolean	AND			29	2	25	3	35	4			2D	4	3D	4*	39	4*	21	6	31	5*							N Z – – – –
	ORA			09	2	05	3	15	4			0D	4	1D	4*	19	4*	01	6	11	5*							N Z – – – –
	EOR			49	2	45	3	55	4			4D	4	5D	4*	59	4*	41	6	51	5*							N Z – – – –
Shift	ASL	0A	2			06	5	16	6			0E	6	1E	7													N Z C – – –
	LSR	4A	2			46	5	56	6			4E	6	5E	7													N Z C – – –
Rotate	ROL	2A	2			26	5	36	6			2E	6	3E	7													N Z C – – –
	ROR	6A	2			66	5	76	6			6E	6	7E	7													N Z C – – –
Miscell.	NOP																					EA	2					– – – – – –
	BRK																					00	7					– – – 1 – –
No. of Bytes		1		2		2		2		2		3		3		3		2		2		1		2		3		

* = 1 Takt mehr bei Seitenübergang;
\+ = 2 Takte ohne Verzweigung, 3 Takte bei Verzweigung, 4 Takte bei Verzweigung mit Seitenübergang.

Beispiele für absolute Adressierung.

```
AD  ØØ  1Ø  LDA $1ØØØ  ;lade den Akku mit (dem Byte) der
                        Speicherstelle $1ØØØ
AE  AA  BB  LDX $BBAA  ;lade das Index-Register X mit
                        Speicherstelle $BBAA
AC  ØØ  ØØ  LDY $ØØØØ  ;lade das Index-Register Y mit
                        Speicherstelle $ØØØØ
8D  1Ø  ØØ  STA $ØØ1Ø  ;speichere A in Speicherstelle $ØØ1Ø
8E  12  34  STX $3412  ;speichere X in Speicherstelle $3412
8C  FF  FØ  STY $FØFF  ;speichere Y in Speicherstelle $FØFF
```

Beispiele für Nullseite-Adressierung.

```
A5  ØØ  LDA $ØØ  ;lade A mit Speicherstelle $ØØ (=$ØØØØ)
A6  1Ø  LDX $1Ø  ;lade X mit Speicherstelle $1Ø
A4  FF  LDY $FF  ;lade Y mit Speicherstelle $FF
85  11  STA $11  ;speichere A in Speicherstelle $11
86  2Ø  STX $2Ø  ;speichere X in Speicherstelle $2Ø
84  DD  STY $DD  ;speichere Y in Speicherstelle $DD
```

Beispiele für unmittelbare Adressierung.

```
A9  ØØ  LDA #ØØ  ;lade A mit dem Byte $ØØ (löscht A)
A2  AØ  LDX #$AØ ;lade X mit dem Byte $AØ
AØ  FF  LDY #$FF ;lade Y mit dem Byte $FF
```

```
                  1    * DEMO: Adressierungsarten
                  2    *
                  3    *------------Felder---------------
                  4    * F1      F2   F3          ;F4
                  5    *-----------------------------------
                  6    START     ORG  $1ØØØ       ;Ursprung
                  7    *
                  8    ZERO1     EQU  $ØØØØ       ;Label
                  9    ZERO2     EQU  $ØØØ1       ;Label
                  1Ø   ZERO3     EQU  $ØØØ2       ;Label
                  11   NUMBER1   EQU  $2ØØØ       ;Label
                  12   NUMBER2   EQU  $2ØØ1       ;Label
                  13   NUMBER3   EQU  $2ØØ2       ;Label
                  14   *
1ØØØ: A5 ØØ       15             LDA  $ØØ         ;Nullseite
1ØØ2: 8D ØØ 2Ø    16             STA  NUMBER1     ;absolut
1ØØ5: A4 Ø1       17             LDY  $Ø1         ;Nullseite
1ØØ7: 8C Ø1 2Ø    18             STY  NUMBER2     ;absolut
1ØØA: A6 Ø2       19             LDX  $Ø2         ;Nullseite
1ØØC: 8E Ø2 2Ø    2Ø             STX  NUMBER3     ;absolut
                  21   *
1ØØF: A9 ØØ       22             LDA  #$ØØ        ;unmittelbar
1Ø11: AA          23             TAX              ;übertragen
1Ø12: A8          24             TAY              ;übertragen
1Ø13: 85 ØØ       25             STA  ZERO1       ;Nullseite
1Ø15: 86 Ø1       26             STX  ZERO2       ;Nullseite
1Ø17: 84 Ø2       27             STY  ZERO3       ;Nullseite
                  28   *
1Ø19: AD ØØ 2Ø    29             LDA  NUMBER1     ;absolut
1Ø1C: AE Ø1 2Ø    3Ø             LDX  NUMBER2     ;absolut
1Ø1F: AC Ø2 2Ø    31             LDY  NUMBER3     ;absolut
1Ø22: 85 ØØ       32             STA  ZERO1       ;Nullseite
1Ø24: 86 Ø1       33             STX  ZERO2       ;Nullseite
1Ø26: 84 Ø2       34             STY  ZERO3       ;Nullseite
1Ø28: 6Ø          35             RTS
```

drei Buchstaben bestehen (obligatorisch). Z. B. ist „LDA" das Mnemonic und „AD" der entsprechende hexadezimale Op-Code für eine der möglichen Adressierungsarten, s. u.

Feld 3: Der „Operand" ist die Adresse oder Speicherstelle, auf die sich der Befehl bezieht (obligatorisch).

Feld 4: „Comment" ist ein Kommentar, der üblicherweise durch ein Semikolon von den vorangehenden Feldern abgegrenzt wird (kann entfallen).

Es sind zwei Typen von Labels zu unterscheiden:

a) „START" und „END" sind z. B. implizite oder interne Labels, die sich auf Speicherstellen innerhalb des Programms selbst beziehen, wobei der Assembler die absoluten Adressen dieser Speicherstellen in Abhängigkeit vom Programmursprung selbst errechnet.

b) „ADDRESS1" und „ADDRESS2" sind z. B. explizite oder externe Labels, die sich auf Adressen beziehen, die außerhalb des eigentlichen Programms liegen. Sie sollten durch EQU-Zeilen zu Beginn des Programms definiert werden.

Ferner ist der Unterschied zwischen einem Label und einem Label-Feld zu beachten. Wie aus Tabelle 3 ersichtlich, kann ein Label sowohl in Feld 1 wie Feld 3 vorkommen. Das gleiche Label, z. B. „ADDRESS1", darf innerhalb desselben Programms beliebig oft in Feld 3 auftauchen, doch darf es nur ein einziges mal in Feld 1 (= Label-Feld) implizit oder explizit definiert werden.

Jedes Assemblerprogramm muß mit einer ORG-Definition des Programmursprungs beginnen und sollte mit dem Mnemonic RTS (= return from subroutine) enden.

7 Elementare Adressierungsarten

Tabelle 4 zeigt, daß der 6502 nur 56 Standard-Mnemonics umfaßt. Ein Befehl wie z. B. „LDA" kann jedoch in verschiedenen Adressierungsformen verwendet werden, so daß die Zahl der Op-Codes erheblich größer ist. Insgesamt gibt es 13 Adressierungsarten, von denen die vier einfachsten im folgenden beschrieben werden.

2.1 Absolute Adressierung

Hier folgt dem Op-Code eine 2-Byte-Adresse in der bereits bekannten, typischen 6502-Form („low byte first").

2.2 Nullseite-Adressierung

Die ersten 256 Bytes des Speichers ($0000 — $00FF) werden, wie ebenfalls

bereits bekannt, Nullseite oder Zero-Page genannt. Bei der Nullseite-Adressierung ist das höherwertige Byte stets $00, d. h. anstelle von $0000 oder $00FF (00000 oder 00255) kann $00 oder $FF (0 oder 255) stehen. Die Nullseite-Adressierung verwendet deshalb immer 1-Byte-Adressen.

2.3 Unmittelbare Adressierung

Die unmittelbare (immediate) Adressierung bezieht sich auf dasjenige Byte, das unmittelbar dem Op-Code folgt, z. B.:

```
1000:    A9 11    LDA # $11
```

Angenommen, die Speicherstelle $1000 enthalte $A9 und die Speicherstelle $1001 enthalte $11. „A9" als Op-Code besagt, daß der Akkumulator mit dem Byte zu laden ist, das unmittelbar auf „A9" folgt, also mit „11". Die unmittelbare Adressierung wird im Quell-Code durch das vorangestellte Nummernzeichen (Doppelkreuz) gekennzeichnet.

2.4 Implizierte Adressierung

Die implizierte (implied) Adressierung bezieht sich auf überhaupt keine Adresse im Speicher, sondern z. B. auf die Register selbst. Wird etwa der Inhalt des Akkumulators in das Index Register X übertragen, werden keinerlei Speicherstellen berührt.

Beispiele für implizierte Adressierung.

```
AA TAX     ;transfers A to X
8A TXA     ;transfers X to A
A8 TAY     ;transfers A to Y
98 TYA     ;transfers Y to A
```

Enthält z.B. A $FF, bewirkt der Befehl TAX, daß $FF nach X übertragen wird. Nach Ausführung von TAX befindet sich der Wert $FF sowohl in A wie auch in X.

Es sei darauf hingewiesen, daß Befehle der Art TXY oder TYX nicht zu dem 6502-Befehlssatz gehören.

Der Befehl

```
60 RTS        ;return from subroutine
```

ist ein weiteres Beispiel der implizierten Adressierung, die dem BASIC-Befehl RETURN oder END entspricht.

Nebenstehende „Demo" faßt die besprochenen Adressierungsarten in einem Gesamtbeispiel zusammen.

8 Tabelle 4: 6502 Mnemonics ...

Die Verwendung dieser Tabelle sei am Beispiel des LDA-Befehls bei der absoluten Adressierung erläutert. In der Spalte „Absol." ist „AD" als Op-Code für das Mnemonic „LDA" zu finden. Rechts von „AD" steht die Zahl „4", die besagt, daß dieser Befehl 4 Mikroprozessor-Takte erfordert. Der 1 MHz 6502 Mikroprozessor schafft 1 Million Takte pro Sekunde, d. h. 1 Takt benötigt 1 μs. Anders formuliert, läßt sich etwa der Befehl LDA $1000 theoretisch 250.000mal pro Sekunde ausführen.

In der letzten Zeile der Tabelle (No. of Bytes) ist in Spalte „Absol." vermerkt, daß dieser Befehl 3 Bytes im Speicher benötigt (1 Byte für den Op-Code und 2 Bytes für die Adresse).

Teil 3

9 Interne und externe Labels

Im vorangehenden Abschnitt wurde bereits zwischen internen und externen Labels unterschieden. Zur Rekapitulation: Ein externes Label bezieht sich auf eine Speicherstelle außerhalb des Assemblerprogramms, während sich ein internes Label auf eine Speicherstelle innerhalb desselben bezieht. Externe Labels werden üblicherweise durch EQU-Gleichungen am Anfang des Assemblerprogramms als absolute Adressen definiert, während die tatsächlichen Adressen interner Labels vom Programmursprung (ORG) abhängen. Interne Labels spielen eine große Rolle bei bedingten Verzweigungen.

```
        ORG   $1000
EXTLBL  EQU   $2000
        LDA   EXTLBL
        CMP   # $00   ; siehe unten
        BEQ   INTLBL; siehe unten
        LDA   # $00
INTLBL  RTS
```

10 Vergleiche und bedingte Verzweigungen

Für jedes Register A, X und Y gibt es je einen Vergleichsbefehl für die verschiedenen Adressierungsarten:

CMP # $FF;	vergleicht A mit $FF
CMP $1000;	vergleicht A mit Speicherstelle $1000
CPX # $FF;	vergleicht X mit $FF
CPX $1000;	vergleicht X mit Speicherstelle $1000

Demo 1

```
                    1          ORG  $1000
                    2   *
                    3   * Indizierte Adressierung
                    4   *
1000: A9 20         5   START  LDA  #$20
1002: 8D 00 20      6          STA  $2000
1005: A9 30         7          LDA  #$30
1007: 8D 00 30      8          STA  $3000
100A: A2 00         9          LDX  #00
100C: A0 00         10         LDY  #00
100E: BD 00 20      11         LDA  $2000,X
1011: 9D 00 40      12         STA  $4000,X     ;4000:20
1014: B9 00 30      13         LDA  $3000,Y
1017: 99 00 50      14         STA  $5000,Y     ;5000:30
                    15  *
                    16  * Inkrement und Dekrement
                    17  *
101A: EE 00 40      18         INC  $4000       ;4000:21
101D: CE 00 50      19         DEC  $5000       ;5000:2F
1020: BD 00 40      20         LDA  $4000,X
1023: E8            21         INX
1024: 9D 00 40      22         STA  $4000,X     ;4001:21
1027: B9 00 50      23         LDA  $5000,Y
102A: C8            24         INY
102B: 99 00 50      25         STA  $5000,Y     ;5001:2F
                    26  *
                    27  * Übertrag (Wrap-around)
                    28  *
102E: A2 FF         29         LDX  #$FF
1030: E8            30         INX              ;X:00
1031: A0 00         31         LDY  #$00
1033: 88            32         DEY              ;Y:FF
                    33  *
                    34  * vergleichen und verzweigen
                    35  *
1034: BD 00 50      36         LDA  $5000,X
1037: C9 30         37         CMP  #$30
1039: F0 02         38         BEQ  EQUAL
103B: D0 03         39         BNE  NOTEQUAL
103D: A9 00         40  EQUAL  LDA  #$00
103F: 60            41         RTS
1040: A9 01         42  NOTEQUAL LDA #$01
1042: 60            43         RTS
```

CPY # $FF; vergleicht Y mit $FF
CPY $1000; vergleicht Y mit Speicherstelle $1000

Auf Vergleiche folgt üblicherweise ein bedingter Verzweigungsbefehl, z.B. BEQ (= branch on equal = verzweige, falls gleich) oder BNE (= branch on not equal = verzweige, falls ungleich), die nachfolgend beschrieben werden.

```
EXAMPLE1 ORG  $1000
EXTLBL   EQU  $2000
         LDA  EXTLBL
         CMP  # $FF
         BEQ  EXIT2
EXIT1    LDA  # $01 ;NOT EQUAL
         RTS
EXIT2    LDA  # $00 ;EQUAL
         RTS
```

In EXAMPLE1 wird A mit dem Byte der Speicherstelle EXTLBL geladen, anschließend wird A mit $FF verglichen. Falls EXTLBL $FF enthält, wird nach EXIT2 verzweigt. Andernfalls fährt das Programm mit EXIT1 fort. Die Befehlsfolge „LDA EXTLBL CMP # $FF BEQ EXIT2" entspricht dem folgenden BASIC-Programm:

```
10 A = PEEK (8192)
20 IF A = 255 THEN GOTO 40
30 PRINT „EXIT1" : END
40 PRINT „EXIT2" : END
```

```
EXAMPLE2 ORG  $1000
EXTLBL1  EQU  $2000
EXTLBL2  EQU  $3000
         LDX  EXTLBL1
         CPX  EXTLBL2
         BNE  EXIT2
EXIT1    LDA  # $00
         RTS
EXIT2    LDA  # $01
         RTS
```

Beim EXAMPLE2 wird EXTLBL1 mit EXTLBL2 verglichen, wobei hier eine Verzweigung nur dann erfolgt, falls EXTLBL1 ungleich EXTLBL2 ist.
Der 6502 läßt auch eine Verzweigung ohne vorausgehenden Vergleich zu. In diesem Fall bedeutet BEQ = verzweige, falls das zuletzt angesprochene Register gleich Null, und BNE = verzweige, falls das Register ungleich Null ist.

```
LDY $2000    oder  LDY $2000
BEQ LABEL1         BNE LABEL2
```

Demo 2

```
                    1          ORG  $1000
                    2   *
                    3   * DEMO 2: $2000-$20FF löschen
                    4   *
                    5   * LOOP1/LOOP2 - Vorwärtsschleifen
                    6   * LOOP3/LOOP4 - Rückwärtsschleifen
                    7   *
                    8   ADDRESS  EQU  $2000
                    9   *
                    10  * LOOP1 löscht $2000-$20FF
                    11  *
1000: A2 00         12         LDX  #$00
1002: A9 00         13         LDA  #$00
1004: 9D 00 20      14  LOOP1  STA  ADDRESS,X
1007: E8            15         INX
1008: E0 00         16         CPX  #$00
100A: D0 F8         17         BNE  LOOP1
                    18  *
                    19  * LOOP2 löscht $2000-$20FF
                    20  * (etwas schneller als LOOP1)
                    21  *
100C: A2 00         22         LDX  #$00
100E: 8A            23         TXA
100F: 9D 00 20      24  LOOP2  STA  ADDRESS,X
1012: E8            25         INX
1013: D0 FA         26         BNE  LOOP2
                    27  *
                    28  * LOOP3 löscht $2000-$20FF
                    29  *
1015: A9 00         30         LDA  #$00
1017: A2 FF         31         LDX  #$FF
1019: 9D 00 20      32  LOOP3  STA  ADDRESS,X
101C: CA            33         DEX
101D: E0 FF         34         CPX  #$FF
101F: D0 F8         35         BNE  LOOP3
                    36  *
                    37  * LOOP4 löscht $2000-$20FF
                    38  * (etwas schneller als LOOP3)
                    39  *
1021: A9 00         40         LDA  #$00
1023: AA            41         TAX
1024: 9D 00 20      42  LOOP4  STA  ADDRESS,X
1027: CA            43         DEX
1028: D0 FA         44         BNE  LOOP4
102A: 60            45         RTS
```

11 Inkrementieren und Dekrementieren

Speicherstellen und Register können inkrementiert (um 1 erhöht) und dekrementiert (um 1 vermindert) werden. Angenommen, der Wert von X sei $10, dann würde der Befehl INX (= inkrementiere X) X von $10 auf $11 erhöhen. Im einzelnen gibt es folgende Inkrement-Dekrement-Befehle:

INC $2000; erhöht Speicherstelle $2000 um 1
DEC $2000; vermindert Speicherstelle $2000 um 1

INX; erhöht X um 1
DEX; vermindert X um 1

INY; erhöht Y um 1
DEY; vermindert Y um 1

Die Befehle „INA" (inkrementiere A) und „DEA" (dekrementiere A) existieren nicht und müssen bei Bedarf simuliert werden:

```
TAY  oder  TAY
INY        DEY
TYA        TYA
```

Was geschieht nun, wenn z.B. Y $FF enthält und inkrementiert oder Y $00 enthält und dekrementiert wird?

```
LDY  # $FF
INY              ; Y = 00
LDY  # $00
DEY              ; Y = FF
```

Im ersten Fall nimmt Y den Wert $00, im zweiten $FF an (Überlauf, „Wraparound").

12 Absolutes, indiziertes Laden und Speichern

Absolute, indizierte Adresse bedeutet: Tatsächliche Adresse = absolute Adresse + Wert des jeweiligen Indexregisters (X oder Y). Beim EXAMPLE1 ist die absolute Adresse EXTLBL = $2000 und der Wert von X = $05, LDA EXTLBL,X entspricht mithin LDA $2005. Bei EXAMPLE2 hat Y den Wert $00, LDA EXTLBL,Y ist also gleichbedeutend mit LDA $2000 (Spezialfall). Mit der absoluten, indizierten Adressierung kann ein Speicherbereich von $100 oder 256 Bytes (= Page = Seite) angesprochen werden.

```
EXAMPLE1 ORG  $1000
EXTLBL   EQU  $2000
         LDX  # $05
         LDA  EXTLBL,X
         RTS

EXAMPLE2 ORG  $1000
EXTLBL   EQU  $2000
         LDY  # $00
         LDA  EXTLBL,Y
         RTS
```

13 Schleifen

Assemblerschleifen entsprechen weitgehend den FOR-NEXT-Schleifen in BASIC. Angenommen, man wollte den Speicherbereich $0300 – $0302 (dezimal 768–770) nach $0303–$0305 (dezimal 771–773) verschieben (= duplizieren), würde dies in BASIC so aussehen:

FOR X = 0 TO 3: POKE 771 + X, PEEK (768 + X): NEXT X

In Assembler wäre folgendes Programm zu schreiben:

```
         ORG  $1000
EXTLBL1  EQU  $0300
EXTLBL2  EQU  $0303
INITLP   LDX  # $00
LOOP     LDA  EXTLBL1,X
         STA  EXTLBL2,X
INCR     INX
         CPX  # $03
         BNE  LOOP
EXIT     RTS
```

0300	0301	0302	0303	0304	0305	vor-
FF	FF	FF	00	00	00	her

0300	0301	0302	0303	0304	0305	nach
FF	FF	FF	FF	FF	FF	her

Zum besseren Verständnis des Programmflusses, wollen wir die Schleifen Schritt für Schritt untersuchen:

LOOP 0: Bei INITLP wird das X-Register initialisiert (X = 0).

LOOP 1: Im ersten Schleifendurchgang ist X = 0, mithin EXTLBL1,X = $0300 und EXTLBL2,X = $0303. Bei INCR wird X um 1 (0 + 1 = 1) erhöht und mit 3 verglichen. Da X noch ungleich 3 ist, wird die BNE-Verzweigung nach LOOP ausgeführt.

LOOP 2: Im zweiten Durchgang entspricht X = 1 und damit EXTLBL1,X = $0301 bzw. EXTLBL2,X = $0304. X wird wiederum um 1 auf jetzt 2 erhöht, ist jedoch weiterhin ungleich 3, so daß erneut die BNE-Verzweigung nach LOOP erfolgt.

LOOP 3: Im dritten und letzten Durchgang gilt X = 2, also EXTLBL1,X = $0302 und EXTLBL2,X = $0305. X wird wiederum um 1 auf jetzt 3 erhöht und mit 3 verglichen. Da X = 3 ist, findet die BNE-Verzweigung jetzt nicht mehr statt und das Programm endet mit RTS.

Demo 1 faßt die neuen Befehle in leichtverständlicher Weise zusammen, während Demo 2 als praktisches Beispiel zeigt, wie man einen Speicherbereich von 256 Bytes löscht, d.h. auf Null setzt.

Teil 4

Im letzten Abschnitt haben wir nur die Vergleiche „gleich" und „ungleich" besprochen. Insgesamt sind folgende Vergleiche möglich:

A = M	BEQ JA
A < > M	BNE JA
A < M	BCC JA
A > = M	BCS JA
A > M	BEQ NEIN
	BCS JA
A < = M	BCC JA
	BEQ JA

A steht für Akkumulator (und ersatzweise für die beiden anderen X- und Y-Register). M steht für Memory, d. h. Speicherstelle oder Vergleichswert.

Aus der Übersicht sind zwei neue Befehle ersichtlich, nämlich BCC und BCS. BCC bedeutet eigentlich Branch on carry clear (verzweige, falls Überlauf-Flag = 0, d. h. gelöscht oder zurückgesetzt ist) und BCS ist die Abkürzung für Branch on carry set (verzweige, falls Überlauf-Flag = 1, d. h. gesetzt ist). Da BCC sachlich für „verzweige, falls kleiner" steht, verwendet man auch oft das Mnemonic BLT (Branch on lower than). Umgekehrt steht BCS sachlich für „verzweige, falls größer oder gleich", so daß sich hierfür das nicht-standardisierte Mnemonic BGE (branch on greater or equal) eingebürgert hat. Befehle in der Art „verzweige, falls größer" und „verzweige, falls kleiner oder gleich" gehören nicht zum 6502-Befehlssatz und müssen deshalb durch Doppelinstruktionen simuliert werden.

Für die praktische Programmierarbeit ist es nach meinen Erfahrungen für den Assembleranfänger besser, wenn er über die sogenannten Status-Flags überhaupt nicht nachdenkt, sondern statt dessen mit der obigen Vergleichstabelle arbeitet. Für die Theoretiker sei jedoch nachfolgend die Funktionsweise des Carry-Flags bei BCC und BCS erläutert:

	1.	2.	3.
A	10	09	10
M	–09	–10	–10
	+01	–01	+00
	C = +	C = –	C = +
	C = 1	C = 0	C = 1

Von den obigen drei Subtraktionen (10–9, 9–10 und 10–10) führt die zweite zu einem Überlauf bzw. zu einem Vorzeichenwechsel. Nehmen wir nunmehr an, daß der 6502-Prozessor bei jedem Vergleich intern eine Subtraktion von A und M vornimmt und das Überlauf-Flag entsprechend setzt, und zwar dergestalt, daß bei einem Vorzeichenwechsel C auf 0 und bei keinem Vorzeichenwechsel C auf 1 gesetzt wird. Dann wird deutlich, weshalb bei den nachfolgenden Beispielen ein Sprung zu den Labels JA bzw. NEIN erfolgt:

```
A < M ?          A > = M ?

LDA # 10         LDA # 10 ;A
CMP # 09         CMP # 09 ;M
BCC NEIN         BCS JA

LDA # 09         LDA # 09
CMP # 10         CMP # 10
BCC JA           BCS NEIN

LDA # 10         LDA # 10
CMP # 10         CMP # 10
BCC NEIN         BCS JA
```

Genau genommen sind die Erläuterungen zu den Subtraktionen zwar praktisch richtig, jedoch theoretisch falsch, weil eigentlich das A vom M subtrahiert und bei 6502-Subtraktionen mit „invertiertem Borgen" gearbeitet wird, wodurch die „normalen" Subtraktionsregeln quasi auf den Kopf gestellt werden. Hierzu schreibt LANCE A. LEVENTHAL in „6502 Assembly Language Subroutines", Berkeley 1982, S. 2: „The Carry Flag acts as an inverted borrow in subtraction. A Compare instruction clears the Carry if the operation requires a borrow and sets it if it does not. Thus the Carry has the opposite meaning after comparison on the 6502 than it has on most other computers."

Da wir gerade bei der Theorie sind, soll kurz auf eine andere Tatsache aufmerksam gemacht werden, die ebenfalls in der Programmierpraxis weitgehend ignoriert werden kann. Betrachten wir zu diesem Zweck den folgenden Programmauszug:

```
                1            ORG  $300
                2     *
                3     * Man beachte bei diesem
                4     * Test die auf BEQ folgende
                5     * hexadezimale Sprungadresse
                6     *
0300: A8        7     L1     TAY
0301: F0 FD     8            BEQ  L1
                9     *
0303: F0 FF     10    L2     BEQ  L2+1
                11    *
0305: F0 FE     12    L3     BEQ  L3
                13    *
0307: F0 00     14           BEQ  L4
0309: A8        15    L4     TAY
                16    *
030A: F0 01     17           BEQ  L5
030C: A8        18           TAY
030D: A8        19    L5     TAY
                20    *
030E: F0 02     21           BEQ  L6
0310: A8        22           TAY
0311: A8        23           TAY
0312: A8        24    L6     TAY
0313: 60        25           RTS
```

```
                1            ORG  $1000
                2     *
                3     * Vergleich-Demo
                4     *
                5     * Welche Stelle muß geändert
                6     * werden, damit das Programm
                7     * bei Branch4 ankommt?
                8     *
1000: A9 A0     9            LDA  #$A0
1002: C9 B0     10           CMP  #$B0
1004: 90 03     11           BCC  BRANCH1
1006: A9 01     12           LDA  #1
1008: 60        13           RTS
1009: C9 B0     14    BRANCH1 CMP #$B0
100B: B0 03     15           BCS  BRANCH2
100D: A9 02     16           LDA  #2
100F: 60        17           RTS
1010: C9 A0     18    BRANCH2 CMP #$A0
1012: F0 03     19           BEQ  BRANCH3
1014: A9 03     20           LDA  #3
1016: 60        21           RTS
1017: C9 B0     22    BRANCH3 CMP #$B0
1019: D0 03     23           BNE  BRANCH4
101B: A9 04     24           LDA  #4
101D: 60        25           RTS
101E: A9 05     26    BRANCH4 LDA #5
1020: 60        27           RTS
```

```
030A: F0 01   LABEL1   BEQ LABEL2
030C: A8               TAY
030D: AA      LABEL2   TAX
```

„F0" ist der Op-Code für BEQ und „A8" bzw. „AA" sind die Op-Codes für TAY bzw. TAX. Nach „F0" steht merkwürdigerweise die hexadezimale Zahl „01". Welche Bedeutung hat sie? Dem Quell-Code ist zu entnehmen, daß ein relativer Vorwärtssprung zum LABEL2 erfolgt. „F0 01" bilden als zusammengehörige Byte-Folge den Befehl BEQ LABEL2. Würde dieser Sprung nicht ausgeführt, weil die Bedingung A = M nicht erfüllt ist, dann würde der prozessorinterne Programmzähler nunmehr an der Speicherstelle $030C angelangt sein. Addieren wir $030C + $01, dann erreichen wir die Speicherstelle $030D, also die Stelle von LABEL2. Mithin können wir festhalten, daß nach dem Vergleichs-Op-Code eine hexadezimale Zahl steht, die die Anzahl der Bytes angibt, um die der Programmzähler versetzt werden muß, damit der BEQ-Sprung an der gewünschten Speicherstelle anlangt. Dieses „Versetzen" wird als Offset bezeichnet.

Was passiert jedoch, wenn kein Vorwärts-, sondern ein Rückwärtssprung stattfinden soll. Betrachten wir hierzu das nachfolgende Beispiel:

```
0300: A8      LABEL1   TAY
0301: F0 FD   LABEL2   BEQ LABEL1
0303: AA               TAX
```

Würde die A = M Bedingung hier nicht erfüllt, dann würde wiederum der nächste Befehl der Speicherstelle $0303 abgearbeitet, nämlich TAX. Was geschieht jedoch, wenn A = M ist. Dann muß ein Rückwärtssprung nach LABEL2 erfolgen. Wenn man mit dem Zählen ab der Speicherstelle $0303 beginnt, gelangt man durch „Bis-drei-Zählen" zur Speicherstelle $0300. Mithin müßte theoretisch in der Speicherstelle $0302 „03" stehen. „03" würde jedoch +3 bedeuten, während hier „FD" −3 bedeutet.

Fassen wir zusammen: Bei relativen Sprüngen steht nach dem Op-Code für BEQ, BNE, BCC und BCS eine entweder positive oder negative hexadezimale Offset-Zahl, um die der Programmzähler ab der der Offset-Zahl folgenden Speicherstelle vorwärts oder rückwärts zum Sprung-Label versetzt werden muß. Hexadezimale Zahlen im Bereich $00−$7F werden als positiv und sinngemäß hexadezimale Zahlen im Bereich $80−$FF als negativ angesehen, und zwar einzig und allein bei den Vergleichsbefehlen:

```
80 —128
 .
 .
 .
FC  —4
FD  —3
FE  —2
FF  —1

00   0
01   1
02   2
03   3
04   4
 .
 .
 .
7F  127
```

Aus der Kurztabelle ist ersichtlich, daß relative Sprünge nur bis 127 Bytes vom Offset-Ursprung entfernt sein dürfen. Diese Beschränkung macht es oft erforderlich, daß man die Programmlogik umdrehen muß, da Vorwärtssprünge über 127 und Rückwärtssprünge über 128 Speicherstellen hinaus vom Assembler als „Bad Branch" zurückgewiesen würden. Dies geschieht durch die Verwendung des absoluten Sprungs JMP, der im nächsten Kapitel beschrieben wird. Im übrigen kann man den Offset ignorieren, da dieser vom Assembler stets automatisch ausgerechnet wird.

Teil 5

In den bisherigen Abschnitten haben wir bereits etwa die Hälfte der 6502-Befehle besprochen, nämlich Lade- und Speicherbefehle (LDA, LDX, LDY; STA, STX, STY), Übertragungsbefehle (TAX, TAY, TXA, TYA), Vergleichsbefehle (CMP, CPX, CPY), bedingte Verzweigungsbefehle (BEQ, BNE, BCC, BCS), Inkrementier- (INC, INX, INY) und Dekrementierbefehle (DEC, DEX, DEY). Ferner wurden die meisten Adressierungsarten behandelt, und zwar absolute Adressierung (LDA $1000), Nullseite-Adressierung (LDA $00), indizierte absolute Adressierung (LDA $1000,X oder LDA $1000,Y), implizite Adressierung (TAY, TXA usw.) sowie die noch nicht expressis verbis benannte relative Adressierung. Letzere ist die Adressierungsart bei allen bedingten Verzweigungsbefehlen (BEQ, BCC usw.).

Trotzdem waren wir bislang nicht in der Lage, „sinnvolle" Programme zu erstellen, weil einige der mächtigeren Befehle noch nicht besprochen wurden. Eine sehr wichtige Gruppe von Befehlen betrifft den sogenannten Stack, auch als Stapel oder „Keller" bezeichnet. Hierzu betrachten wir das nachfolgende erweiterte Modell des 6502-Prozessors:

6502				
X-Register	A-Register	Y-Register	Stapelzeiger	Programmzähler
LL	LL	LL	LL	LL HH

Die jeweils 1 Byte großen Indexregister X und Y sowie der Akkumulator als allgemeines Datenregister sind bereits bekannt. Der Programmzähler ist ein 2 Bytes umfassendes Register und enthält die Adresse der Speicherstelle des jeweils als nächstes vom Prozessor zu bearbeitenden Befehls. Beispiel:

```
1000: AD 00 20 LDA $2000
1003: 8D 00 30 STA $3000
```

Nehmen wir an, der Prozessor sei bereits bei Speicherstelle $1000 angelangt. Dann enthält der Programmzähler die Adresse LL – HH, Low Byte – High Byte, 00 – 10 = $1000, d.h. der Prozessor ist auf die Adresse des Op-Codes AD = LDA gerichtet. Dieser Befehl wird in Verbindung mit den darauffolgenden 2 Adreßbytes interpretiert und dann entsprechend ausgeführt. Anschließend wird der Programmzähler auf Speicherstelle $1003 erhöht. Bei bedingten (relativen) Verzweigungen wird er mit der relativen Adresse geladen, während er bei unbedingten (absoluten) Verzweigungen mit der absoluten Adresse der Speicherstelle, zu der der Sprung erfolgen soll, geladen wird. Es gibt zwei Arten von absoluten Verzweigungen oder Sprüngen:

JMP = Jump (entspricht GOTO in BASIC)
JSR = Jump Subroutine (entspricht GOSUB in BASIC)

RTS = Return from Subroutine (entspricht RETURN in BASIC)
RTS = entspricht als letztes RTS eines Assemblerprogramms END in BASIC

Betrachten wir zunächst folgendes Programm:

```
1000: 4C 00 20   JMP $2000
1003: 60         RTS
 .
 .
 .
2000: 4C 03 10   JMP $1003
```

Im ersten Schritt wird der Programmzähler mit der Adresse $1000 geladen. Dann holt sich der Prozessor den Op-Code 4C = JMP, lädt den Programmzähler mit der darauffolgenden Adresse $2000 und springt zu $2000. Dort holt sich der Prozessor den Op-Code 4C, lädt den Programmzähler mit der nachfolgenden Adresse $1003 und springt zu $1003. Hier steht nun RTS, das als letzter Op-Code das Assemblerprogramm beendet.

Was geschieht aber mit dem Programmzähler, wenn statt eines JMP ein JSR gewünscht wird. Sehen wir dazu JSR-Beispiel 1 an:

JSR-Beispiel 1

```
1000: 20 00 20   JSR $2000
1003: 60         RTS
 .
 .
 .
2000: 60         RTS
```

JSR-Beispiel 2

```
0FFE: A2 FF      LDX #$FF
0FFF: 9A         TXS
1000: 20 00 20   JSR $2000
1003: 60         RTS
 .
 .
 .
2000: A9 AA      LDA #$AA
2002: 48         PHA
2003: 68         PLA
2004: 60         RTS
```

Zunächst wird der Programmzähler mit $1000 geladen. Dann holt sich der Prozessor den Op-Code 20 = JSR und lädt den Programmzähler mit der Adresse $2000, zu der gesprungen werden soll. Da der Programmzähler jedoch nur jeweils eine einzige Adresse enthalten kann, erhebt sich die Frage, wie sich der Prozessor „merken" kann, daß er nach RTS = RETURN bei Adresse $2000 wieder zu derjenigen Adresse zurückfindet, die auf den Befehl JSR $2000 folgt. Zu diesem Zweck unterhält der Prozessor einen 256 Bytes umfassenden Stapel, der bei allen 6502-Mikrocomputern den Speicherbereich $0100 bis $01FF einnimmt. Dieser Stapel wird rückwärts, d.h. ab $01FF, mit den RTS-Rücksprungadressen gefüllt. Neben dem Stapel gibt

Beispiel

```
:ASM
                  1            ORG  $1000
                  2    *
                  3    * Addition und Subtraktion
                  4    * ========================
                  5    *
                  6    * HEXOUT zeigt Wert in A-Register
                  7    * als hexadezimale Zahl an.
                  8    * HEXOUT ändert nur das A-Register
                  9    *
                 10    * CHROUT zeigt Character in
                 11    * A-Register als ASCII-Zeichen an.
                 12    * CHROUT läßt alle drei Register
                 13    * A, X und Y unverändert.
                 14    *
                 15    * Beides sind ROM-Firmware-Routinen
                 16    * des Apple IIc/IIe/II Plus
                 17    *
                 18    HEXOUT   EQU  $FDDA       ;Apple
                 19    CHAROUT  EQU  $FDED       ;Apple
                 20    *
                 21    * 8-Bit-Addition ohne Überlauf
                 22    *
                 23    * 10 + 10 = 20             hex $14
                 24    *
1000: 18         25            CLC
1001: A9 0A      26            LDA  #10
1003: 69 0A      27            ADC  #10
1005: 20 DA FD   28            JSR  HEXOUT
1008: 20 86 10   29            JSR  KOMMA
                 30    *
                 31    * 8-Bit-Subtraktion ohne "Unterlauf"
                 32    *
                 33    * 10 - 10 = 0              hex $00
                 34    *
100B: 38         35            SEC
100C: A9 0A      36            LDA  #10
100E: E9 0A      37            SBC  #10
1010: 20 DA FD   38            JSR  HEXOUT
1013: 20 86 10   39            JSR  KOMMA
                 40    *
                 41    * 16-Bit-Addition mit Überlauf-
                 42    * prüfung.
                 43    *
                 44    * $4044 + $3033 = $7077
                 45    *
1016: 18         46            CLC
1017: A9 44      47            LDA  #$44        ;Low
1019: 69 33      48            ADC  #$33        ;Low
101B: 8D 9A 10   49            STA  LL
101E: A9 40      50            LDA  #$40        ;High
1020: 69 30      51            ADC  #$30        ;High
1022: 8D 9B 10   52            STA  HH
1025: 90 03      53            BCC  OKAY1
1027: 20 8C 10   55            JSR  ERROR
102A: AD 9B 10   56    OKAY1   LDA  HH          ;High
102D: 20 DA FD   57            JSR  HEXOUT
1030: AD 9A 10   58            LDA  LL          ;Low
1033: 20 DA FD   59            JSR  HEXOUT
1036: 20 86 10   60            JSR  KOMMA
                 61    *
                 62    * 16-Bit-Subtraktion mit
                 63    * "Unterlauf"-Prüfung
                 64    *
                 65    * $4044 - $3033 = $1011
                 66    *
1039: 38         67            SEC
103A: A9 44      68            LDA  #$44        ;Low
103C: E9 33      69            SBC  #$33        ;Low
103E: 8D 9A 10   70            STA  LL
1041: A9 40      71            LDA  #$40        ;High
1043: E9 30      72            SBC  #$30        ;High
1045: 8D 9B 10   73            STA  HH
1048: B0 03      74            BCS  OKAY2
104A: 20 8C 10   75            JSR  ERROR
104D: AD 9B 10   76    OKAY2   LDA  HH          ;High
1050: 20 DA FD   77            JSR  HEXOUT
1053: AD 9A 10   78            LDA  LL          ;Low
1056: 20 DA FD   79            JSR  HEXOUT
1059: 20 86 10   80            JSR  KOMMA
                 81    *
                 82    * Verschiedene fehlerhafte
                 83    * Additionen und Subtraktionen
                 84    *
                 85    * Nach SEC 3 + 1 = 5!      hex $05
                 86    *
105C: 38         87            SEC
105D: A9 03      88            LDA  #3
105F: 69 01      89            ADC  #1
1061: 20 DA FD   90            JSR  HEXOUT
1064: 20 86 10   91            JSR  KOMMA
                 92    *
                 93    * Nach CLC 3 - 1 = 1!      hex $01
                 94    *
1067: 18         95            CLC
1068: A9 03      96            LDA  #3
106A: E9 01      97            SBC  #1
106C: 20 DA FD   98            JSR  HEXOUT
106F: 20 86 10   99            JSR  KOMMA
                100    *
                101    * überlauf 255 + 1 = 0!    hex $00
                102    *
1072: 18        103            CLC
1073: A9 FF     104            LDA  #255
1075: 69 01     105            ADC  #1
1077: 20 DA FD  106            JSR  HEXOUT
107A: 20 86 10  107            JSR  KOMMA
                109    *
                110    * "Unterlauf" 0 - 1 = 255! hex $FF
                111    *
107D: 38        112            SEC
107E: A9 00     113            LDA  #0
1080: E9 01     114            SBC  #1
1082: 20 DA FD  115            JSR  HEXOUT
1085: 60        116            RTS
                117    *
                118    * Unterroutine zum Anzeigen
                119    * eines Kommas
                120    *
1086: A9 AC     121    KOMMA   LDA  #","
1088: 20 ED FD  122            JSR  CHAROUT
108B: 60        123            RTS
                124    *
                125    * Unterroutine zum Anzeigen
                126    * des Wortes "Error:"
                127    *
108C: A2 00     128    ERROR   LDX  #0
                129    *
                130    * 6-stelliges Wort "Error:"
                131    *
108E: BD 9C 10  132    ERROR1  LDA  ERRWORT,X
1091: 20 ED FD  133            JSR  CHAROUT
1094: E8        134            INX
1095: E0 06     135            CPX  #6
1097: D0 F5     136            BNE  ERROR1
1099: 60        137            RTS
                138    *
                139    * Der Pseudo-Opcode HEX
                140    * reserviert eine Speicherstelle
                141    * mit einem Hexwert.
                142    * Der Pseudo-Opcode ASC
                143    * reserviert einen Speicherraum
                144    * mit einem String.
                145    *
109A: 00        146    LL      HEX  00
109B: 00        147    HH      HEX  00
109C: C5 F2 F2  148    ERRWORT ASC  "Error:"
109F: EF F2 BA

--End assembly--

162 bytes

Errors: 0

Symbol table - alphabetical order:

   CHAROUT =$FDED     ERROR   =$108C     ERROR1  =$108E     ERRWORT =$109C
   HEXOUT  =$FDDA     HH      =$109B     KOMMA   =$1086     LL      =$109A
   OKAY1   =$102A     OKAY2   =$104D

Symbol table - numerical order:

   OKAY1   =$102A     OKAY2   =$104D     KOMMA   =$1086     ERROR   =$108C
   ERROR1  =$108E     LL      =$109A     HH      =$109B     ERRWORT =$109C
   HEXOUT  =$FDDA     CHAROUT =$FDED
```

Stapel									Zeiger	Programm
100 ...	1F8	1F9	1FA	1FB	1FC	1FD	1FE	1FF		
										Beispiel 1
1.	00	00	00	00	00	00	00	00	FF	Initialisierung
2.	00	00	00	00	00	00	02	10	FD	JSR $2000
3.	00	00	00	00	00	00	02	10	FF	RTS
										Beispiel 2
1.	00	00	00	00	00	00	00	00	??	LDX #$FF
2.	00	00	00	00	00	00	00	00	FF	TXS
3.	00	00	00	00	00	00	02	10	FD	JSR $2000
4.	00	00	00	00	00	00	02	10	FD	LDA #$AA
5.	00	00	00	00	00	AA	02	10	FC	PHA
6.	00	00	00	00	00	AA	02	10	FD	PLA
7.	00	00	00	00	00	AA	02	10	FF	RTS

es den Stapelzeiger, der als eine Art X-Register aufgefaßt werden kann und der beim Einschalten des Mikrocomputers oder beim Reinitialisieren des Prozessors in der Regel auf $FF ($0100,X) gesetzt wird.

Nehmen wir an, der Mikrocomputer sei gerade eingeschaltet worden, der Stack sei noch leer und der Stapelzeiger sei auf $FF initialisiert. Ferner sei der Programmzähler auf $1000 eingestellt, wo der Befehl JSR $2000 stehen soll. Wie bereits bekannt, wird der Programmzähler mit der Adresse $2000 geladen. Gleichzeitig wird die Rücksprungadresse auf den Stapel geschoben (to push on stack), und zwar nicht die eigentliche Adresse $1003, sondern $1003 – $01 = $1002. High Byte $10 wird in $01FF und Low Byte $02 in $01FE im Stapel zwischengespeichert. *Vor* jeder Zwischenspeicherung wird der Stapelzeiger um 1 vermindert und erst *danach* das HH- oder LL-Byte auf den Stack geschoben. Der Prozessor führt also quasi intern ein Programm aus, das wir wie folgt simulieren können:

```
DEX          ;decrement
LDA #$10     ;push
STA $0100,X
DEX          ;decrement
LDA #$02     ;push
STA $0100,X
```

Wenn der Prozessor nunmehr bei Speicherstelle $2000 auf RTS stößt, zieht er die Rücksprungadresse vom Stapel, erhöht sie um $01 und überträgt sie in den Programmzähler. Beachten Sie folgenden Merksatz:

Decrement before pushing!
Increment before pulling!

Neben JSR gibt es noch andere 6502-Befehle, die den Stapel sowie den Stapelzeiger beeinflussen.

PHA (= Push accumulator on stack) dekrementiert den Stapelzeiger und schiebt den Inhalt des Akkumulators auf den Stack.

PLA (= Pull accumulator from stack) inkrementiert den Stapelzeiger, zieht den Inhalt der entsprechenden Stapelspeicherstelle vom Stapel und überträgt ihn in den Akkumulator.

Mit der Befehlsfolge PHA ... PLA läßt sich der Akkumulatorinhalt vorübergehend zwischenspeichern, doch präge man sich die goldene Regel ein, daß der PLA stets *vor* dem nächsten RTS stattfinden muß, andernfalls würde RTS den Akkumulatorinhalt anstelle der Rücksprungadresse vom Stapel ziehen!

TSX (Transfer stack pointer to X register) überträgt den Inhalt des Stapelzeigers in das X-Register.

TXS (Transfer X register to stack pointer) überträgt den Inhalt des X-Registers in das Stapelzeigerregister.

Dabei ist zu beachten, daß der Stapelzeiger nur über das X-Register, also z.B. nicht über den Akkumulator, gelesen und verändert werden kann. Mit der Befehlsfolge

```
LDX #$FF
TXS
```

läßt sich der Stapelzeiger neu initialisieren (siehe JSR-Beispiel 2).

Teil 6

Bevor wir uns der hexadezimalen Addition und Subtraktion zuwenden, werfen wir zunächst einen Blick auf die dezimale Addition, wie sie ein „ABC-Schütze" durchführt (Kasten 1).

Fall 1 erfordert keinen Übertrag. Fall 2 verlangt einen Übertrag in der ersten Stelle. In Fall 3 sind zwei Überträge nötig; gleichzeitig sehen wir, daß die Addition von zwei s-stelligen Zahlen höchstens zu einem s + 1-stelligen Ergebnis führt.

Betrachten wir nunmehr die hexadezimale Addition (Kasten 1). Zur Erinnerung: Die hexadezimalen Ziffern erstrecken sich von $0 bis $F, womit ein Übertrag von $1 erst dann notwendig wird, wenn die Summe von zwei hexadezimalen Ziffern $F (= dezimal 15) überschreitet.

In Fall 1 ist wiederum kein Übertrag nötig, denn $1 + $1 = $2 und $D + $1 = $E. In Fall 2 ist ein Übertrag in der ersten Stelle erforderlich, denn $F + $1 = $10. In Fall 3 sind zwei Überträge vonnöten. Wir addieren hier bei der ersten Stelle: $F + $1 = $0 plus $1 „im Sinn", und bei der zweiten Stelle $E + $1 + $1 „im Sinn" = $0 + $1 „im Sinn", was in die dritte Stelle übertragen wird. Auch für die hexadezimale Addition gilt, daß die Summe von zwei s-stelligen Hexzahlen höchstens s + 1-stellig sein kann.

Das Carry-Flag ist das Übertrags-Flag oder Übertrags-Bit. Mit dem Befehl CLC (Clear Carry) kann der Übertrag gelöscht und mit SEC (Set Carry) gesetzt werden. Mit dem Befehl ADC (= Add with Carry) werden zwei zweistellige Hexzahlen zusammenaddiert und mit SBC (= Subtract with Carry) voneinander subtrahiert. Für eine 8-Bit-Addition (= Addition von zwei zweistelligen Hexzahlen), die höchstens eine zweistellige Hexzahl als Ergebnis haben soll, und für die 8-Bit-Subtraktion gelten stets die in Kasten 2 wiedergegebenen Prozeduren.

Die hexadezimale Addition und Subtraktion geschieht stets byte-weise und nicht nibble-weise (halbbyte-weise). Zum Beispiel besteht die zweistellige Hexzahl $A0 aus zwei Nibbles oder zwei Halbbytes $A und $0. Sollen nur zwei niederwertige Nibbles, z.B. $A und $B, addiert werden, so muß dies in der Form

```
CLC
LDA   #$0A   $0 High Nibble,
             $A Low Nibble
ADC   #$0B   $0 High Nibble,
             $B Low Nibble
```

vor sich gehen, d.h. die höherwertigen Nibbles sind auf Null zu setzen. Bei den zwei Beispielen in Kasten 3 werden bei der Addition jeweils zwei Nibbles zu einem Byte zusammengefaßt.

Kasten 1

```
Fall 1     Fall 2     Fall 3                                  Fall 1     Fall 2     Fall 3
  7 1        7 9        9 9                                     D 1        D F        E F
  1 1        1 1        9 9                                     1 1        1 1        1 1
             1          1 1     Übertrag („Carry", „im Sinn")                1          1 1     Carry
-------    -------    -------                                 -------    -------    -------
  8 2        9 0        1 9 8                                   E 2        F 0        1 0 0
                       (Überlauf in 3. Stelle)                                       (Überlauf in 3. Stelle)
```

Dezimale Addition **Hexadezimale Addition**

Kasten 2

```
CLC                 Vor der Addition das Carry löschen
LDA   HEXZAHL1      Summand1 in A-Register laden
ADC   HEXZAHL2      Summand2 zum A-Register hinzuzählen
BCS   FEHLER        Überlauf, falls Summe > $FF
BCC   OKAY          Kein Überlauf, falls Summe ≤ $FF
```

1-Byte-Addition

```
SEC                 Vor der Subtraktion Carry setzen
LDA   HEXZAHL1      Minuend in A-Register laden
ADC   HEXZAHL2      Subtrahend vom A-Register abziehen
BCC   FEHLER        Unterlauf, falls Differenz < $00
BCS   OKAY          Kein Unterlauf, falls Differenz ≥ $00
```

1-Byte-Subtraktion

Kasten 3

```
      EF    EF    $00EFEF =    61423                    FF    FF    $00FFFF =    65535
      11    11    $001111 =     4369                    FF    FF    $00FFFF =    65535
   01    01                                          01    01
-------------                                     -------------
01    01    00    $010100 =    65792              01    FF    FE    $01FFFE =   131070
                         - 65536 =   $010000                                  - 65536 =   $010000
                             256 =   $000100                                    65534 =   $00FFFE
```

Beispiele für 2-Byte-Additionen mit Übertrag in drittes Byte

Kasten 4

```
CLC                 ;4stellige Addition
LDA   SUMMAND1LL    ;Low Byte 1. Summand
ADC   SUMMAND2LL    ;Low Byte 2. Summand
STA   SUMMELL       ;Low Byte Summe
LDA   SUMMAND2HH    ;High Byte 1. Summand
ADC   SUMMAND2HH    ;High Byte 2. Summand
STA   SUMMEMM       ;Middle Byte Summe
BCC   OKAY          ;Summe 4stellig!
ADC   #$00          ;$00 + Carry 1 addieren
STA   SUMMEHH       ;High Byte Summe; 6stellig!
```

2-Byte-Addition

```
SEC                 ;4stellige Subtraktion
LDA   MINUENDLL     ;Low Byte Minuend
SBC   SUBTRAHLL     ;Low Byte Subtrahend
STA   DIFFERLL      ;Low Byte Differenz
LDA   MINUENDHH     ;High Byte Minuend
SBC   SUBTRAHHH     ;High Byte Subtrahend
STA   DIFFERMM      ;Middle Byte Differenz
BCS   OKAY          ;Differenz 4stellig!
SBC   #$00          ;$00 – Carry 0 subtrahieren
STA   DIFFERHH      ;High Byte Differenz; 6stellig!
```

2-Byte-Subtraktion

Wenn b Bytes umfassende Hexzahlen addiert bzw. subtrahiert werden sollen und falls das Ergebnis b + 1 Bytes umfassen darf, ist die Prozedur aus Kasten 4 anzuwenden. Man beachte, daß stets mit dem niederwertigsten Byte zu beginnen ist.
Das hier abgedruckte ausführliche Beispiel ist unser erstes, etwas größeres Assemblerprogramm. Es verwendet systemspezifische Routinen. HEXOUT (hexadezimaler Output) zeigt den Inhalt des Akkumulators als 2stellige Hexzahl an. OUT (Charakter Output) gibt das im Akkumulator befindliche Byte als ASCII-Zeichen aus. Ferner werden zwei Pseudo-Opcodes HEX und ASC verwendet, die bei verschiedenen Assemblern verschiedene Bezeichnungen haben können. Schließlich wird erstmals am Ende des Listings eine „Symbol Table" abgedruckt, die die im Programm benutzten Labels alphabetisch und numerisch, d.h. adreßmäßig, sortiert zusammengestellt.

Im einzelnen werden folgende Beispiele durchgerechnet:

8-Bit-Addition/Subtraktion ohne Überlauf/Unterlauf: Eine Überprüfung des Überlaufs/Unterlaufs läßt sich sparen, wenn im voraus bekannt ist, daß aufgrund der Werte keiner möglich ist.

16-Bit-Addition/Subtraktion mit nicht erwartetem, aber möglichem Überlauf/Unterlauf: Hier soll das Ergebnis im Bereich $0000 – $FFFF liegen, also mithin 4stellig oder 2 Byte lang sein. Sofern dieser Bereich bei Addition über- bzw. bei Subtraktion unterschritten wird, führt dies zu einer Fehlermeldung.

Verschiedene fehlerhafte 8-Bit-Additionen/Subtraktionen: Addition nach SEC, Subtraktion nach CLC, Überlauf und Unterlauf. Man beachte, daß ein Mikroprozessor im Gegensatz zu BASIC, PASCAL usw. keine „freundlichen" Fehlermeldungen wie „OVERFLOW ERROR", „OUT OF RANGE", „ILLEGAL QUANTITY" usw. ausgibt, wenn tatsächliche Fehler in einem Maschinenprogramm auftreten. Vielmehr tut der Prozessor so, als wäre nichts gewesen.

Teil 7

14 Binärzahlen

Im letzten Teil wollen wir uns mit einigen Bitbefehlen befassen. Ein Mikroprozessor weiß intern noch nichts von dezimalen und hexadezimalen Zahlen, sondern kennt nur Binär- oder Dualzahlen, d.h. die Zustände Spannung ein – Spannung aus. Eine Binärzahl ist eine Positionszahl mit der Basis 2, der zur Verdeutlichung in der Regel ein Prozentzeichen vorangestellt wird.

Beispiele für zweistellige Binärzahlen:

%00 = dezimal 0
%01 = dezimal 1
%10 = dezimal 2
%11 = dezimal 3
usw.

Die vierstellige Binärzahl %1111 errechnet sich wie folgt:

0. Stelle = 2 hoch 0 * 1 = 1
1. Stelle = 2 hoch 1 * 1 = 2
2. Stelle = 2 hoch 2 * 1 = 4
3. Stelle = 2 hoch 3 * 1 = 8

ergibt zusammen 1 + 2 + 4 + 8 = dezimal 15 = hexadezimal $F.

Ein Byte kann als eine achtstellige Binärzahl dargestellt werden, wobei die acht Bitstellen von rechts nach links und von 0 bis 7 durchnumeriert sind, z. B.:

76543210 Bitposition
%11110000 Bitmuster

Die rechten vier Bits, also Bit 0 – 3, bilden das niederwertige Halbbyte, die linken vier, Bit 4 – 7, das höherwertige. Ein Bit kann gesetzt (= 1) oder zurückgesetzt (= 0) sein. Die Summe der Bits wird auch als Bitmuster bezeichnet. Im Gegensatz zu höheren Programmiersprachen wie BASIC, Pascal usw. eignen sich die Befehle eines Mikroprozessors hervorragend zu Bitmanipulationen. Die Befehle AND und ORA dienen zum Setzen und Zurücksetzen (Wegblenden, Löschen) von Bitmustern, während die Befehle ASL und LSR zum Verschieben von Bitmustern verwendet werden.

15 AND und ORA

AND = logischer Und-Befehl (and)
ORA = logischer Oder-Befehl (or A)

AND undiert den Inhalt des Akkumulators A mit einem Bitmuster der Speicherstelle M und überträgt das Resultat der Undierung in den Akkumulator. Nehmen wir an, der Akkumulator A enthalte den Wert %11111111 und die Speicherstelle M das Bitmuster %00001111. Dann gilt:

```
LDA #%11111111
AND MEMORY
```

%11111111 = A
%00001111 = M
%00001111 = A

AND setzt bei A diejenigen Bits, die zuvor sowohl bei A als auch bei M gesetzt waren. Es gilt nachstehende Wahrheitswerttabelle:

A	M	A
1	1	1
1	0	0
0	1	0
0	0	0

ORA setzt bei A diejenigen Bits, die zuvor mindestens bei entweder A oder M gesetzt waren.

```
LDA #%11111111
ORA MEMORY
```

%11111111 A
%00001111 M
%11111111 A

Für ORA gilt die Wahrheitswerttabelle:

A	M	A
1	1	1
1	0	1
0	1	1
0	0	0

AND dient damit zum Wegblenden unerwünschter und ORA zum setzen gewünschter Bits. Nehmen wir an, wir wollten bei einem im Arbeitspeicher abgelegten, 256 Bytes umfassenden Text Bit 7 jedes Bytes setzen. Zu diesem Zweck würde folgende Schleife genügen:

```
      LDY #0
LOOP  LDA MEMORY, Y
      ORA %10000000
      STA MEMORY, Y
      INY
      BNE LOOP
```

Wollten wir umgekehrt Bit 7 bei allen Bytes wegblenden, müßten wir so programmieren:

```
      LDY #0
LOOP  LDA MEMORY, Y
      AND %01111111
      STA MEMORY, Y
      INY
      BNE LOOP
```

Man beachte, daß AND und OR nur den Inhalt von A und niemals den von M direkt beeinflussen, so daß zur Änderung einer Speicherstelle M zunächst M in A geladen und nach der Bitmanipulation wieder in M zurückgespeichert werden muß.

16 ASL und LSR

ASL = Linksverschiebung (arithmetic shift left)
LSR = Rechtsverschiebung (logical shift right)

ASL verschiebt wahlweise das Bitmuster von A oder von M insgesamt um eine Stelle nach links. Ferner wird Bit 7 in das Carry-Flag (= Übertrags-Flag) übertragen und Bit 0 auf 0 gelöscht. Sofern das Bitmuster von A selbst verändert werden soll, spricht

```
                  1              ORG  $1000
                  2     *
                  3     * BINAER-ANZEIGE
                  4     * ===============
                  5     *
                  6     * Anzeige eines hexadezimalen
                  7     * Bytes als 8stellige Binaerzahl
                  8     * Das anzuzeigende Byte muss
                  9     * sich bereits im Akkumulator
                  10    * befinden.
                  11    *
                  12    *
1000: 4C 05 10    13             JMP  BEGINN
                  14    *
                  15    * ROM-Routine CHAROUT zeigt
                  16    * beim Apple // Inhalt des
                  17    * Akkumulators als ASCII-
                  18    * zeichen an. A-, X- und
                  19    * Y-Register werden nicht
                  20    * veraendert.
                  21    *
                  22    CHAROUT  EQU  $FDED
                  23    *
1003: FF          24    HEXZAHL  HEX  FF         ;spaeter 0
1004: 08          25    BITS     HEX  08         ;8 Bits
                  26    *
                  27    * Hexzahl zwischenspeichern
                  28    * und Bitzaehler auf 8 setzen
                  29    *
1005: 8D 03 10    30    BEGINN   STA  HEXZAHL
1008: A9 08       31             LDA  #8
100A: 8D 04 10    32             STA  BITS
                  33    *
                  34    * Momentanes Bitmuster laden
                  35    * nach links schieben und das
                  36    * dadurch veraenderte Bit-
                  37    * muster erneut speichern
                  38    *
100D: AD 03 10    39    SCHLEIFE LDA  HEXZAHL
1010: 0A          40             ASL
1011: 8D 03 10    41             STA  HEXZAHL
                  42    *
                  43    * Falls Carry-Bit 0, "0" anzeigen
                  44    * Falls Carry-Bit 1, "1" anzeigen
                  45    *
1014: 90 04       46             BCC  NULL
1016: A9 B1       47    EINS     LDA  #"1"
1018: D0 02       48             BNE  ANZEIGE    ;stets
101A: A9 B0       49    NULL     LDA  #"0"
                  50    *
101C: 20 ED FD    51    ANZEIGE  JSR  CHAROUT
                  52    *
                  53    * Bitzaehler um 1 vermindern.
                  54    * Wenn noch nicht alle 8 Bits
                  55    * abgearbeitet wurden, erneut
                  56    * Schleife durchlaufen.
                  57    *
101F: CE 04 10    58             DEC  BITS
1022: D0 E9       59             BNE  SCHLEIFE
1024: 60          60             RTS

--End assembly--

37 bytes

Errors: 0
```

man von der sogenannten Akkumulator-Adressierung.

Beispiele:

LDA #%01111111 ;A vorher
ASL A ;A nachher

76543210 Bitposition
A %01111111 vor ASL
A %11111110 nach ASL
(Carry-Flag jetzt 0)

LDA #%10101010 ;A vorher
ASL A ;A nachher

76543210 Bitposition
A %10101010 vor ASL
A %01010100 nach ASL
(Carry-Flag jetzt 1)

LSR verschiebt wahlweise das Bitmuster von A oder M insgesamt um eine Stelle nach rechts. Ferner wird Bit 0 in das Carry-Flag übertragen und Bit 7 gelöscht.

Beispiele:

LDA #%01111111 ;A vorher
LSR A ;A nachher

76543210 Bitposition
A %01111111 vor LSR
A %00111111 nach LSR
(Carry-Flag jetzt 1)

LDA #%10101010 ;A vorher
LSR A ;A nachher

76543210 Bitposition
A %10101010 vor LSR
A %01010101 nach LSR
(Carry-Flag jetzt 0)

ASL und LSR werden unter anderem bei Multiplikationen und Divisionen benötigt. So bewirkt ein einzelner ASL-Befehl eine Verdopplung des vorherigen Akkumulator-Inhalts (A = A * 2), z. B.:

LDA #%00001010 ;dezimal 10
ASL A ;A jetzt %00010100 = dezimal 20

Umgekehrt bewirkt ein einzelner LSR-Befehl eine Division durch 2 (A = A : 2), z. B.:

LDA #%00011110 ;dezimal 30
LSR A ;A jetzt %00001111 = dezimal 15

Nebenstehendes Beispiel zeigt, wie man eine zweistellige Hexadezimalzahl als achtstellige Binärzahl anzeigen kann.

APPLE -- DISKETTEN LAUFWERKE -- APPLE

Original Disk II m. Controller + DOS 3.3 + Hdbuch	DM	995,—
Original Disk II (2. Laufwerk)	DM	825,—
Duo-Disk Station Slimline, 2 x 143KB Chinon	DM	995,—
Siemens Disk-Laufw., 143KB, m. Kabel + Gehäuse	DM	525,—
Abor-Disk-Laufw., Slimline, 143KB, m. Kab. + Geh.	DM	398,—
Chinon Slimline, 143KB, Superlinse, m. Kab. + Geh.	DM	495,—
Teac 55F, 1 MB unf. Kapazität, Shugartbus	DM	598,—
Teac 55F, kpl. in Gehäuse, 40/80 Track, 1MByte anschlußfertig, mit Umschaltung 40/80 + Kabel	DM	648,—
Zweitlaufwerk für Apple II C, 143 KB, mit Kabel	DM	498,—

APPLE -- INTERFACES + MAINBOARDS -- APPLE

Disk Controller f. 2 Original o. Kompat. Drives	*DM	89,—
Super-Controller f. 2x Teac 55F, mit Software	**DM	188,—
80 Zeich-Karte mit 64K RAM + Softsw. f. IIe	DM	198,—
80 Zeich-Karte II, m. Softswitch, 2 Zeichensätze	*DM	159,—
16K RAM Erweiterung für II und kompatible	*DM	98,—
Z 80A Interface Karte für CPM 2.2	*DM	88,—
Z 80B Interface Karte für CPM 3.0, m. 64K RAM	DM	595,—
Printer-Grafik Interface, Epson kompatible	*DM	98,—
Centronic-Paralell Text-Interface Karte	*DM	98,—
Printer-Grafik Interface, NEC/ITOH kompatibel	*DM	169,—
Anschluß Kabel f. Paralell + Grafik Interface	DM	34,—
Buffer Grafik-Interface, benutzbar für Drucker Epson/Nec/Itoh/Okidata u. andere m. 32K Buffer	DM	348,—
Buffer Interface wie vor aber mit 64K Buffer	**DM	595,—
Anschluß Kabel f. Buffer Interface Karte	DM	39,—
232C Serielle Interface Karte	*DM	109,—
Super Serielle Interface Karte, Fullduplex	**DM	188,—
Sprach (Speech) Karte f. Sprachwiedergabe	*DM	78,—
6522 Paralell Interface Karte	*DM	149,—
Clock Karte (Datum/Uhrzeit) Ein/Ausgabe	*DM	129,—
Eprom Writer Karte (2716 - 2764)	*DM	129,—
IEEE-488 Interface Karte	*DM	398,—
Logo-Karte mit Diskette und Handbuch	DM	498,—
Musik Karte m. Diskette und Handbuch	*DM	119,—
PAL Color Interface Karte (UHF + Video)	*DM	129,—
RGB Interface Karte (f. Apple II + II+)	*DM	169,—
Wild Karte (kopiert über RAM-Bereich)	*DM	119,—
128K RAM Erweiterungs Karte m. Patchsoftware	**DM	398,—
256K RAM Erweiterungs Karte m. Patchsoftware	**DM	598,—
6809 Prozessor Exell-9 Interfacekarte	+ DM	398,—
IC-Tester Interface Karte (RAMS/TTL 54/74)	DM	448,—
Hauptplatine 48K, o. Firmware Eproms, 8 Slots	DM	448,—
Hauptplatine 64K, wie vor	DM	498,—

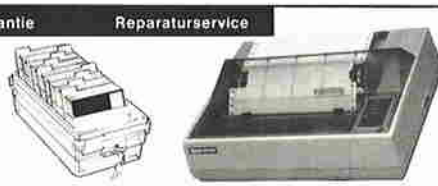
APPLE -- LEERLATINEN -- APPLE

Leerplatinen der obigen Interface Karten sind alle in vergoldet, m. Bestück-Aufdruck + Best.-Plan lieferbar.

Leerplatinen mit der Kennzeichnung	**DM	39,—
Leerplatinen mit der Kennzeichnung	*DM	24,50
Leerplatinen mit der Kennzeichnung	+ DM	33,—
Experimentier Platine, für Apple Slot EX300	DM	29,90
Leerplatine Motherboard, 48K, mit Best.-Aufdruck	DM	66,—
Leerplatine Motherboard, 64K, mit 6502 + Z80	DM	99,—

100% Apple-kompatibel bei Verwendung des Apple-Betriebssystems.

6 Mon. Garantie **Reparaturservice**



APPLE -TASTATUREN+LEERGEHÄUSE+NETZTEILE- APPLE

Standard Einbau-Tastatur, ASCII, freibel. 9 Tasten, Doppelbelg. aller Tasten, Groß+Kleinschr. Nr. N26	DM	139,—
Tastatur wie vor, jedoch mit 15er Block Nr. N67	DM	178,—
Tastatur Nr. N67, mit sep. Gehäuse, kpl. m. Kabel	DM	249,—
Seperate Tastatur IBM-Look, anschlußfertig mit Kabel, Dopp. belegt. Tasten, freiprog. Funkt. Tasten	DM	298,—
Seperate Tastatur, Eprom progr.-bar, Cursorblock Tastenfeld mit 24 Funktionen, Deutsch o. ASCII	DM	398,—
Leergehäuse Standard, passend f. Tastatur Nr. N26	DM	98,—
dto. wie vor, jedoch passend für 15er Tast. N67	DM	119,—
Leergehäuse wie Mewa 9000, für Einbau von zwei Slimline-Laufwerken + Tastaturanschluß, Plastic	DM	159,—
Leergehäuse wie vor, jedoch in Metall-Ausführung	DM	198,—
Leergehäuse IBM-Look, f. 2 Laufw. Standard 5"	DM	198,—
Leergehäuse für 1 Slimline Laufwerk, Metall	DM	19,90
Leergehäuse f. Duo-Disk = 2x Slimline-Laufwerke	DM	98,—
Leergehäuse f. Duo-Disk = 2x Standard-Laufw.	DM	109,—
Leergehäuse f. Duo-Disk = 2x Slimline + Netzteil	DM	179,—

Schaltnetzteile für APPLE u. kompatible Rechner

+5V/3,5A -5V/0,5A +12V/2A -12V/0,5A	N73	DM	99,80
+5V/5 A -5V/0,5A +12V/2,5A -12V/0,5a	N74	DM	119,80
+5V/7,5A -5V/0,5A +12V/2A -12V/0,5A	N75	DM	149,50

-------- -- DISKETTEN + DISKETTEN-BOXEN -- --------

Profi Disk Box, m. Schloß, Klars.-Deckel, 100 Disks	DM	49,—
Disk Box mit Klars.-Deckel, ca. 70 Disketten 5"	DM	26,—
Hartbox für 10 Disketten, mit Klappdeckel	DM	5,90

Disketten, mit Verstärkungsring, 3 Jahre Garantie

ABX 100 = SS/SD 10 Stck. a. 3,98	50 Stck. a. 3,88	500 Stck. a. 3,59	
ABX 200 = SS/DS 10 Stck. a. 4,08	50 Stck. a. 3,98	500 Stck. a. 3,69	
ABX 400 = DS/DD 10 Stck. a. 4,98	50 Stck. a. 4,88	500 Stck. a. 4,59	

APPLE -- ORIGINAL + KOMPATIBLE - COMPUTER -- APPLE

APPLE IIe, 64KRAM, Ascii-Tastat. + UHF-Modut.	DM	2198,—
APPLE IIe, Einstiegspaket = Computer 64KRAM, + Philips-Monitor 18 Mhz grün + Siemens F122 - Laufwerk 143K m. Controller + Lerndiskette	DM	3098,—
APPLE II C, 128 KRAM, ASCII Tastatur	DM	2795,—
APPLE II C, wie vor, jedoch deutsche Tastatur	DM	2998,—
MACINTOSH System komplett m. Mouse+Tastatur + Macwrite/Macpaint, Systemguide-Diskette	DM	5995,—
MEWA-48K, Apple kompat., Netzt. 5Amp, Tastatur mit Dopp-Belegung + 10 frei Prog. Tasten, mit UHF-Mod., ohne Firmware Eproms.	DM	698,—
MEWA-64K, wie vor jedoch mit 15er Block	DM	728,—
MEWA 9000er Serie, mit seper. Tastat. DIN o. ASCII m. dopp. belegt. Funkt-Tasten, Cursorbl. + 15er Block Netzt. 5 Amp., Gehäuse für 2x Slimline-Laufwerke UHF-Modulator, o. Firmw.-Proms (12KROM), 48K RAM	DM	998,—
MEWA 9000-64-C, wie vor jedoch 64K RAM	DM	1098,—
MEWA 9000-64-C Einstiegspaket, wie vor, jedoch mit 1 Disk Drive eingeb. + Disk-Contr. + Monitor grün	DM	1898,—
IBM-Look Gehäuse für MEWA 9000 Serie Aufpreis	DM	39,—

-------- -- MATRIX- UND TYPENRAD - DRUCKER -- --------

CP-80 Matrix Drucker, Epson kompat., vollgrafik f.	DM	739,—
CP-80 X mit eingeb. Interface für VC 64 f. Sim. B.	DM	899,—
SPEEDY 100-80, Epson kompat., 100 Z/s, v.-grafikf.	DM	759,—
GEMINI 10X, Neu mit Textsp., 100 Z/s., 9x9 Mat.	DM	998,—
GEMINI 15X, wie 10X, jedoch bis 375 mm Papierbr.	DM	1198,—
ITOH 8510 B, Die Zuverlässigkeit in Person	DM	1475,—
JUKI 6100, Typenraddr., 22 Z/s, TA-Typenrader	DM	1495,—

IBM * KOMPATIBEL ***** IBM *** KOMPATIBEL ***

Alle Teile unterliegen einer sorgfältigen Endkontrolle und wir übernehmen daher volle Garantie für die Funktionsfähigkeit, sowohl für die gelieferten Leerplatinen, als auch für die fertig bestückten Boards und Interface Karten, die fast ohne Ausnahme mit gesockelten IC's geliefert werden.

IBM -- Kompatible Interface Karten und Mainboards -- IBM

Disk Controller Karte für 2 Disk-Drives	*KL-2020	DM	228,-
Color Video Board.	*KL-2050	DM ab	398,-
Monochrome Video Boards	*KL-2060	DM ab	328,-
RS 232 Drucker Interface Karte	KL-2074	DM	199,-
Centronics Paralell Interface Karte	KL-2072	DM	148,-
Multifunktion Karte, RAM-Bereich, RS232, Paralell, Clock, mit O K bestückt	*KL-2040	DM	398,-
dto. wie vor, jedoch mit 128K bestuckt	*KL-2041	DM	548,-
dto. wie vor, jedoch mit 256K bestuckt	*KL-2042	DM	698,-
512K RAM Karte, mit O K bestuckt	KL-2095	DM	248,-
dto. wie vor, mit 128K bestuckt	KL-2096	DM	398,-
Multi I/O Interface Board, RS232, Disk-Controller, Centronics, Clock, Joystick und Lightpen-Port	*KL-2071	DM	568,-
Eprom Writer Karte (bis 128K benutzbar)			598,-
Hardisk/Winchester Host-Adapter Karte	*KL-2022	DM ab	998,-
Mainboard XT Version, 8 Slots 0 bestückt mit 1 Boot-Eprom, 6 Eprom-Plätze frei	*KL-1004	DM	699,-
dto. wie vor, jedoch mit 128K bestuckt	KL-1005	DM	849,-
dto. wie vor, jedoch mit 256K bestuckt	*KL-1006	DM	999,-
Mainboard PC Version, 8 Slots, 64K bestückt mit 1 Boot-Eprom, 6 Eprom-Plätze frei	*KL-1010	DM	896,—
dto. wie vor, jedoch mit 128K bestuckt	*KL-1011	DM	996,—
dto. wie vor, jedoch mit 256K bestuckt	*KL-1012	DM	1296,—

Bei den mit * gekennzeichneten Artikeln gehört zum Lieferumfang ein Manual, und z. T. ein Schaltbild.

IBM -- Kompatible Leerplatinen + Boards -- IBM

Disk Controller f. 2 Drives	DM	58,—
Color Grafik Video Board	DM	78,—
RS232 Interface Karte	DM	68,—
Centronics Paralell Interface Karte	DM	68,—
Multifunktion Interface Board	DM	99,—
Eprom Writer Interface Karte	DM	78,—
Multi I/O Board	DM	78,—
Mainboard XT Version, 8 Slots	DM	98,—
Mainboard PC Version, 8 Slots	DM	96,—
RAM Card für Speicher-Erweiterung	DM	68,—

Alle Leerplatinen werden mit Bestückungsplan geliefert. Schaltbilder, soweit verfügbar, Lieferung gegen Aufpreis.

IBM -- Gehäuse/Netzteile/Tastaturen/Diskdrives -- IBM

Rechner Leergehäuse f. Einbau v. 2 Drives	DM	236,—
Profi-Tastatur, m. Kabel im Gehause, ASCII	DM	395,—
dto. wie vor, jedoch deutsche DIN Belegung	DM	489,—
Standard Tastatur, IBM-Look, ASCII	DM	298,—
Netzteil, mit Ventilator, 100 Watt	DM	299,—
dto. wie vor jedoch 135 Watt	DM	338,—
Diskelten Laufwerk, 2x40 Track DS/360 KB	DM	498,—
Harddisk 10MByte, kpl. m. Xebec Controller	DM	2990,—

Sonderliste gegen Rückporto. Katalog gegen 3 DM in Briefmarken

Computer-Artikel Nachnahmeversand unfrei. Zwischenverkauf vorbehalten.
Angebote freibleibend unter Anerkennung unserer Lieferbedingungen. Technische Änderungen vorbehalten. *Apple ist eingetragenes Warenzeichen der Fa. Apple-Computer Inc., Kalifornien. Ware mit Rückgaberecht, besonders gekennzeichnet, muß frei zurückgeschickt werden. *IBM* ist eingetragenes Warenzeichen der Firma IBM GmbH Frm.

COMPUTER CENTER CONEX
5650 SOLINGEN 11 · Postfach 11 02 06 - 7 T
Telefon (0221) 75449

+

ERICH-WILLI MEYER
6343 FROHNHAUSEN · Postfach 70 11
Telefon (0 27 71) 3 50 71

INTUS – Lernprogramme

Demo-Disk DM 10,–

- mit 8 voll lauffähigen Teilprogrammen zum interaktiven Lernen. Dazu 7 Denkspiele.
- Lauffähig auf Apple IIe, IIc, teilweise II+. Katalog mit über 200 Programmen gratis.
- Lernprogramme für Sprache, Mathe, Basic, Maschinenschreiben, Informatik, Schnellesen, Vorschule, usw.

INTUS SOFTWARE

Kaiserstraße 21 · 7890 Waldshut-Tiengen · Telefon 07751-7920

Telefon (02 41) 3 49 62
Noppiusstraße 19, 5100 Aachen

RÖCKRATH MICROCOMPUTER

Nach dem ROM-Listing jetzt das DOS-3.3-Listing für den Apple-II

- sehr ausführlich englisch kommentierter Assemblerquellcode mit Symbolen
- Als Quellcode für Toolkit-Assembler geeignet
- Crossreferenzliste aller benutzter Symbole
- DOS-Tips (Autostart, Leserfehler)
- Programme (Init ohne DOS, DOS-Checker)

Außerdem folgende Bücher vorrätig: (ISBN 3-925074-..-.)

ROM-Listings für Apple-II	(ISBN -04-X)	59,–	DM
TRS-80 Model I, Genie I+II	(ISBN -01-5)	69.50	DM
TRS-80 Model III	(ISBN -02-3)	79,–	DM
Colour-Genie	(ISBN -03-1)	59,–	DM
Dragon-32/64-Lexikon	(ISBN -05-8)	69,–	DM
Maschinensprache für TRS-80/Genie und CG	(ISBN -07-4)	49,–	DM
Assemblertricks auf dem Colour-Genie	(ISBN -08-2)	49,–	DM

F. Engels: Apple-II-DOS-3.3-Assembler-Listing (107 S., DIN-A5, kart., ISBN 3-925074-09-0, 59,– DM)

Apple und IBM kompatible Computer

16K, Z80, Diskcontroler je	110,–
80 Zeichenkarte mit Softswitch 2 Zeichensätze	195,–
Motherboard 48K ohne Firmware	610,–
Erphi-controler mit Autopatch	300,–
Siemenslaufwerk F 122	515,–
Philips X3134 2×80 Track	465,–
TEAC FD-55B 2×40 Track	468,–
TEAC FD-55F 2×80 Track	565,–
Neu: Stardrucker SG 10	**920,–**
Monochrome Monitore	ab 375,–
Farbmonitore	ab 998,–
Tastaturen für IBM und Apple	ab 330,–

Versand nur per Nachnahme oder Vorkasse

Weiteres Zubehör für Apple und IBM gegen frankierten Rückumschlag.

Preissenkung:
128K Karte (Saturn kompatibel) **395,–**

Zusatzkarten und Motherboard ausnahmslos deutsche Fertigung mit ausgesuchten Bauteilen.

Ulf Mohwinkel Electronic
Berliner Straße 73
5090 Leverkusen Fettehenne
Telefon 02 14/9 37 81

APPLE-INTERFACE

Weitgehend deutsche Produktion

MESON II-PC, 48K, Apple-kompatibel	ab 1000,–
Z80, 16K	85,– 95,–
PAL, EPROM, CENTRONICS, RS 232	je 170,–
80 Zeichen SoftSwitch	200,–
ADD 2B VIA-Card (6522) mit RAM u. BackUp	170,–
ADD 4B, PIA-Card (6821) mit RAM u. BackUp	150,–
JOYPORT zum Anschluß von zwei Atari-Joystick	40,–
TEAC FD 55 F	550,–
Shugart SA 465 80 Track DS sehr leise	550,–
r-phi-Controller AFDC4 Autopatch	300,–
LDD 103, 40 Track Apple-kompatibel Slim	420,–

Preise pro Stck., inkl. MwSt.

Ladenverkauf: Mo. 12–18 Uhr, Di.-Do. 15–18 Uhr, Sa. 10–13 Uhr
Telefon 030-833 13 03 (wie Ladenverkauf)

TOMBSTONE-MICRO

T. Tank & G. Körber · Gardeschützenweg 72 · 1000 Berlin 45

KLEINTEILE

Stand 29. 5. 1985

UPD 4116-200ns	1–9	10+
UPD 4164-150ns	3,30	3,90
MM 4164 Ungef.	6,20	7,50
TMM 4164 Ungef.	27,50	26,50
6116 LP3 (SRM 2016)	10,50	9,—
2708-450ns	7,—	5,—
2716-450ns	13,30	13,—
2732-450ns	16,—	15,50
2764-250ns	14,—	13,—
27128-250ns	21,—	19,—
27256-250ns	55,—	50,—
280A CPU	6,70	6,20
74 LS 00	—,60	—,80
74HC 00	2,20	2,—
LED 5mm Rot superhell 10 Stck.	1,80	1,50
LED 8 mm Rot	—,90	—,80
LED 5 mm Rot blink.	—,75	—,65
LED 5 mm Grün o. Gelb blink.	—,90	—,80

Weitere Bauteile und Messgeräte an A. Minosettbestellwert 50,– DM, sonst + 10,— DM, Porto und Verpackung: bis 250 g 5,— DM. Versand per NN oder VK, bei NN + 1,50 DM

APPLE //e
APPLE][
und kompatible Computer

Universeller EPROM-Programmer 4003

■ Programmiert alle gängigen EPROM-Typen (z.B.: 2716, 2732,-64,-128,2508,-16,-32,-64...) ■ Voll menügesteuerte Software auf Diskette ■ Kein Schalten, Löten oder Stecken nötig ■ Programmierspannung wird im Gerät erzeugt ■ Verbindung zum Rechner über Flachbandkabel ■ Rote und grüne Leuchtdiode zur Betriebsart-Anzeige ■ Komplett mit 28 poligem Textool-Sockel ■

Fertiggerät DM 269,50 ■ Bausatz + Anleitung DM 219,–

APPLE //e 80 Zeichen Karte +64 KByte RAM

■ 80 gestochen scharfe und flimmerfreie Zeichen / Zeile
■ plus zusätzliche 64 KByte schnelle dynamische RAM's
■ ermöglicht Double Hires Graphik (560 * 192 Punkte)
■ vergoldete Steckerleiste und Qualitätsbauteile

Geprüfte Platine + Demo Disk + Beschreibung DM 219,50
Bausatz DM 189,– ■ Leerplatine + Bauanleitung DM 59,–
Ab Lager lieferbar. Alle Preise inklusive Mehrwertsteuer

DOBBERTIN
INDUSTRIE-ELEKTRONIK
Brahmsstraße 9, 6835 Brühl, Tel. (06202) 71417

Fortsetzung von Seite 31

und normale Anzeige, Ctrl-Q kehrt in das Hauptmenü zurück.

QUITTIEREN – Die Eingabe von „Q" beendet das Programm. Das Programm BITEDITOR steht weiterhin im Rechnerspeicher und kann geändert werden, ohne den ebenfalls im Speicher enthaltenen Zeichensatz-File zu beeinflussen.

Die vier wichtigsten Variablen und ihre Werte werden in der untersten Bildschirmzeile des Hauptmenüs angezeigt und können gegebenenfalls in der Initialisierungs-Routine des Programms geändert werden.

In **Bild 3** ist der Buchstaben B noch einmal fett und daneben fett-invers dargestellt.

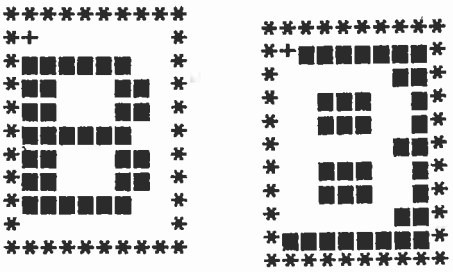


Bild 3

Das Programm erklärt sich durch die reichlich verwendeten REMs praktisch von selbst. Sie können die REMs beim Abschreiben des Programms weglassen oder mit einem REM-Remover-Programm entfernen, da Zeilen, in denen nur Erläuterungen stehen, nicht angesprungen werden. Das zu bearbeitende Zeichen wird als Binärmuster in einen Array geschrieben, welches als Matrix am Bildschirm zu sehen ist. Änderungen erfolgen im Array. Erst bei der Übernahme durch <RETURN> wird der geänderte Array in den Speicher geschrieben.

Ich habe die Änderung an mehreren Apple-Arbeitsplätzen durchgeführt, je nach Monitorqualität mit inversem, fettem und normalem, nicht fettem Zeichensatz oder mit beiden Zeichensätzen fett. Je billiger der Monitor, desto größer der sichtbare Erfolg. Unser Fernsehgerät mit Video-Eingang läßt die 80-Zeichendarstellung überhaupt erst nach der Modifikation zu, mit nur geringfügig schlechterer Qualität läuft diese auch über einen guten HF-Modulator und den Antenneneingang eines TV-Gerätes.

Selbst bei teuren 80-Zeichenkarten wird der inverse Zeichensatz scheinbar nur durch Invertieren des normalen gewonnen. Dabei ist es allerdings fast unmöglich – auch bei sehr guten Monitoren –, beim Betrieb am Apple eine Einstellung zu finden, die zugleich eine gute normale und eine gute inverse Darstellung erlaubt. Mich wundert es deshalb, daß der inverse Zeichensatz auf den 80-Zeichenkarten nicht fett oder in ähnlicher Form geliefert wird. Kompromisse müssen bei einer Matrix mit relativ niedriger Auflösung wohl immer gemacht werden, eine so deutliche Steigerung der Lesbarkeit wie hier rechtfertigt die Modifikation aber unbedingt.

Anmerkung zur Sammeldiskette:
Auf der Peeker-Sammeldiskette sind die drei Zeichensatz-Files NORMAL, FETT und FETT.INVERSE enthalten.

BITEDITOR

```
1000 REM BITEDITOR:
1010 REM zeigt Speicherinhalte binär an und erlaubt
1020 REM diese zu editieren (z.B. Zeichensätze)
1030 REM Klamt 11.83 Version 10.84
1040 GOTO 4620: REM Programmbeginn
1100 REM ************************************************
1110 REM DEZ -> HEX Umwandlung
1120 H$ = ""
1130 HH = DEC / 16:DEC = INT (HH):HH = (HH - DEC) * 16: IF
     HH > 9 THEN HH = HH + 7
1140 H$ = CHR$ (HH + 48) + H$: IF DEC < > 0 THEN 1130
1150 RETURN
1200 REM ************************************************
1210 REM HEX -> DEZ Umwandlung
1220 DEC = 0: FOR I = 1 TO LEN (H$):HH = ASC ( MID$
     (H$,I,1)) - 48: IF HH > 9 THEN HH = HH - 7
1230 DEC = DEC * 16 + HH: NEXT I: RETURN
1300 REM ************************************************
1310 REM Array füllen und DEZ -> BIN Umwandlung
1320 DEC = DEC / 256
1330 FOR J = 0 TO WI:DEC = DEC * 2:AR(J,I) = INT (DEC):DEC
     = DEC - INT (DEC)
1340 NEXT : RETURN
1400 REM ************************************************
1410 REM Eine Array-Zeile lesen, BIN -> DEZ Umwandlung
1420 DEC = 0
1430 FOR J = 0 TO WI
1440 DEC = DEC * 2 + AR(J,I)
1450 NEXT : RETURN
1500 REM ************************************************
1510 REM Eine Zeile des Bitmusters ausgeben
1520 HTAB HS
1530 FOR J = 0 TO WI
1540 VTAB VS + I
1550 BIN = AR(J,I)
1560 IF BIN = 0 THEN PRINT " ";
1570 IF BIN = 1 THEN INVERSE : PRINT " ";: NORMAL
1580 NEXT : PRINT : RETURN
1600 REM ************************************************
1610 REM Adresse ausgeben
1620 IF WW THEN RETURN
1630 FOR I = 0 TO LG - 1
1640 VTAB VS + I
1650 IF NOT W AND I > 0 THEN 1710
1660 DEC = ST + I: IF DEC < R1 OR DEC > R2 THEN PRINT CHR$
     (7);: GOTO 1710
1670 GOSUB 1120: REM DEZ -> HEX
1680 IF LEN (H$) < 4 THEN FOR K = 1 TO 4 - LEN (H$):H$ =
     "0" + H$: NEXT
1690 HTAB HA: PRINT "$";H$: REM Adresse ausgeben
1700 NEXT
1710 RETURN
1800 REM ************************************************
1810 REM Inhalte ausgeben
1820 FOR I = 0 TO LG - 1
1830 IF ST + I < R1 OR ST + I > R2 THEN PRINT CHR$ (7);:
     GOTO 1900
1840 VTAB VS + I
1850 GOSUB 1420: REM Array lesen, BIN -> DEZ Umwandlung
1860 GOSUB 1120: REM DEZ -> HEX
1870 IF LEN (H$) = 1 THEN H$ = "0" + H$
1880 HTAB HC: PRINT "$";H$: REM Eine Zeile ausgeben
1890 NEXT
1900 RETURN
2000 REM ************************************************
2010 REM Bitmuster-Fenster neu schreiben
2020 POKE 33,WI + 1: POKE 32,HS - 1: POKE 34,VS - 1: POKE
     35,VS + LG - 1: HOME : TEXT : REM Fenster löschen
2030 FOR I = 0 TO LG - 1
2040 IF ST + I < R1 OR ST + I > R2 THEN PRINT CHR$ (7);:
     RETURN : REM Bereichskontrolle
2050 DEC = PEEK (ST + I)
2060 GOSUB 1320: REM Array füllen
2070 GOSUB 1520: REM Bitmuster ausgeben
2080 NEXT : RETURN
2100 REM ************************************************
2110 REM Nach rechts verschieben
2120 GOSUB 4310: REM 'Bitte warten'
2130 FOR I = 0 TO LG - 1
2140 DEC = PEEK (I + ST)
2150 GOSUB 1320: REM DEZ -> BIN
2160 FOR J = 0 TO WI
2170 AR(WI - J + 1,I) = AR(WI - J,I): REM nach rechts
2180 NEXT
2190 AR(0,I) = AR(WI + 1,I): REM rechte Reihe wird linke
     Reihe
```

```
2200 GOSUB 1420: REM BIN -> DEZ
2210 POKE I + ST,DEC: REM Geändertes Array in Speicher
     schreiben
2220 NEXT
2230 A$ = "N": GOTO 2490
2300 REM ***************************************************
2310 REM Alle Zeichen invertieren
2320 GOSUB 4310: REM 'Bitte warten'
2330 FOR I = 0 TO FL
2340 NR = PEEK (I + FS)
2350 NR = 255 - NR: REM invertieren
2360 POKE I + FS,NR: REM In Speicher schreiben
2370 NEXT
2380 A$ = "N": GOTO 2490: REM Neustart
2400 REM ***************************************************
2410 REM Kommandoeingabe
2420 X = HS:Y = VS: REM Cursor-Position
2430 GOSUB 3620: REM CS$ Cursor setzen
2440 POKE - 16368,0: REM Keyboard-Strobe zurücksetzen
2450 GOSUB 4210: VTAB VW: HTAB HW: GET A$
2460 IF A$ = CHR$ (86) THEN 2120: REM Verschieben nach
     rechts
2470 IF A$ = CHR$ (8) AND ST - IV < R1 THEN PRINT CHR$
     (7);: GOTO 2440: REM Bereichskontrolle
2480 IF A$ = CHR$ (21) AND ST + IV > R2 THEN PRINT CHR$
     (7);: GOTO 2440: REM Bereichskontrolle
2490 IF A$ = CHR$ (8) OR A$ = CHR$ (21) THEN GOSUB 2920:
     REM Adreßfenster löschen
2500 IF A$ = "W" AND W = 1 THEN GOSUB 2920: REM
     Adreßfenster löschen
2510 IF A$ = CHR$ (8) OR A$ = CHR$ (21) OR A$ = CHR$ (32)
     OR A$ = "W" OR A$ = "N" THEN GOSUB 2820: REM Fenster
     mit Inhalten löschen
2520 IF A$ = "L" THEN 4020: REM Laden
2530 IF A$ = "S" THEN 4120: REM Speichern
2540 IF A$ = "I" THEN 3420: REM Cursor hoch
2550 IF A$ = "J" THEN 3440: REM Cursor links
2560 IF A$ = "K" THEN 3460: REM Cursor rechts
2570 IF A$ = "M" THEN 3480: REM Cursor abwärts
2580 IF A$ = CHR$ (32) THEN AR(X - HS,Y - VS) = NOT AR(X -
     HS,Y - VS): GOTO 2430: REM Leertaste = konvertieren
2590 IF A$ = CHR$ (13) THEN GOSUB 4310: GOTO 3320: REM
     <RETURN>, in Speicher schreiben
2600 IF A$ = CHR$ (84) THEN 3920: REM Test 80 Zeichen
2610 IF A$ = "N" THEN GOSUB 4310: GOSUB 1620: GOSUB 2020:
     GOTO 2420: REM Neustart
2620 IF A$ = CHR$ (9) THEN 2320: REM Alle Zeichen
     invertieren
2630 IF A$ = CHR$ (8) THEN GOSUB 4310:ST = ST - IV: GOSUB
     1620: GOSUB 2020: GOSUB 4550: GOTO 2420: REM
     Vorhergehende Seite
2640 IF A$ = CHR$ (21) THEN GOSUB 4310:ST = ST + IV: GOSUB
     1620: GOSUB 2020: GOSUB 4550: GOTO 2420: REM Nächste
     Seite
2650 IF A$ = "Q" THEN RETURN : REM Programm beenden
2660 IF A$ = "W" THEN W = 1: GOSUB 1620: GOSUB 1820:W =
     0:WW = 1: HTAB HW: VTAB VW: GOTO 2440: REM Werte
     anzeigen
2670 IF (A$ < "0" OR A$ > "9") AND (A$ < "A" OR A$ > "F")
     THEN PRINT CHR$ (7);: GOSUB 4210: GOTO 2440: REM
     Eingabe ist Hex-Zahl
2680 GOTO 3020: REM Neue Hex-Adresse
2800 REM ***************************************************
2810 REM Fenster mit Inhalt löschen
2820 POKE 33,3: POKE 32,HC - 1: POKE 34,VS - 1: POKE 35,21:
     HOME : TEXT : RETURN
2900 REM ***************************************************
2910 REM Fenster mit Adressen löschen
2920 POKE 33,5: POKE 32,HA - 1: POKE 34,VS - 2: POKE 35,21
2930 HOME : TEXT :WW = 0: RETURN
3000 REM ***************************************************
3010 REM Neue Hex-Adresse
3020 GOSUB 4210: PRINT "NEUE HEX-ADRESSE: $";
3030 B$ = A$: GOTO 3130
3040 GET A$
3050 IF A$ = CHR$ (8) AND B$ = "" THEN PRINT CHR$ (7);:
     GOTO 3040
3060 IF A$ = CHR$ (8) AND LEN (B$) = 1 THEN B$ = "": GOTO
     3130
3070 IF A$ = CHR$ (8) THEN B$ = LEFT$ (B$, LEN (B$) - 1):
     GOTO 3130
3080 IF A$ = CHR$ (13) THEN 3150
3090 IF (A$ < "0" OR A$ > "9") AND (A$ < "A" OR A$ > "F")
     THEN PRINT CHR$ (7);: GOTO 3040
3100 IF LEN (B$) < 5 THEN B$ = B$ + A$: PRINT A$;: GOTO
     3040
3110 PRINT CHR$ (7);: GOTO 3040
3120 REM **********
3130 VTAB 22: HTAB 19: CALL - 868: PRINT B$;: GOTO 3040
3140 REM **********
3150 IF B$ = "" THEN 2440: REM Kommandoeingabe
3160 GOSUB 4310:H$ = B$: GOSUB 1220: REM HEX -> DEZ
     Umwandlung
3170 IF DEC < R1 OR DEC > R2 THEN PRINT CHR$ (7);: GOTO
     3230: REM Bereichskontrolle
3180 ST = DEC
3190 GOSUB 2920: GOSUB 2820: REM Fenster löschen
3200 GOSUB 1620: REM Adresse ausgeben
3210 GOSUB 2020: REM Bitmuster-Fenster neu schreiben
3220 GOSUB 4210: REM Zeile 22 und 23 löschen
3230 GOTO 2420: REM Kommandoeingabe
3300 REM ***************************************************
3310 REM Array in Speicher schreiben
3320 FOR I = 0 TO LG - 1: GOSUB 1420: POKE ST + I,DEC: NEXT
3330 GOTO 2440: REM Kommandoeingabe
3400 REM ***************************************************
3410 REM I J K M Editiermodus
3420 IF Y = VS THEN PRINT CHR$ (7);: GOTO 2450
3430 GOSUB 3720:Y = Y - 1: GOSUB 3620: GOTO 2450
3440 IF X = HS THEN PRINT CHR$ (7);: GOTO 2450
3450 GOSUB 3720:X = X - 1: GOSUB 3620: GOTO 2450
3460 IF X = HS + WI THEN PRINT CHR$ (7);: GOTO 2450
3470 GOSUB 3720:X = X + 1: GOSUB 3620: GOTO 2450
3480 IF Y = VS + LG - 1 THEN PRINT CHR$ (7);: GOTO 2450
3490 GOSUB 3720:Y = Y + 1: GOSUB 3620: GOTO 2450
3600 REM ***************************************************
3610 REM CS$-Cursor setzen
3620 VTAB Y: HTAB X: GOSUB 3810
3630 IF CS = 0 THEN PRINT CS$
3640 IF CS = 1 THEN INVERSE : PRINT CS$: NORMAL
3650 RETURN
3700 REM ***************************************************
3710 REM Cursor löschen
3720 VTAB Y: HTAB X: GOSUB 3810
3730 IF CS = 0 THEN PRINT " "
3740 IF CS = 1 THEN INVERSE : PRINT " ": NORMAL
3750 RETURN
3800 REM ***************************************************
3810 CS = AR(X - HS,Y - VS): RETURN
3900 REM ***************************************************
3910 REM 80-Zeichen-Test
3920 PRINT : PRINT D$"PR#3": PRINT
3930 PRINT CHR$ (12);
3940 PRINT "↑I(NVERSE DARSTELLUNG  ↑N(ORMAL  ↑Q(UITTIEREN"
3950 GET A$: PRINT A$;: IF A$ =  CHR$ (17) THEN  PRINT
     CHR$ (26)"1";: GOTO 2440: REM Test 80 Zeichen beenden
3960 IF A$ = CHR$ (14) THEN PRINT CHR$ (26)"2";: REM
     Normale Anzeige
3970 IF A$ = CHR$ (9) THEN PRINT CHR$ (26)"3";: REM Inverse
     Anzeige
3980 GOTO 3950
4000 REM ***************************************************
4010 REM File laden
4020 GOSUB 4210: PRINT "FILE-NAME LADEN? ";: INPUT "";F$
4030 IF F$ = "" THEN 4070
4040 GOSUB 4310
4050 PRINT D$;"BLOAD";F$;",A$";FS$
4060 GOSUB 2820: GOSUB 2920: GOSUB 1620: GOSUB 2020: GOSUB
     4210: GOTO 2420
4070 GOSUB 4210: GOTO 2440
4100 REM ***************************************************
4110 REM File speichern
4120 GOSUB 4210: PRINT "FILE-NAME SPEICHERN? ": PRINT
     "VORGABE: ";F$;: VTAB 22: HTAB 21: INPUT "";FF$
4130 IF FF$ < > "" THEN F$ = FF$
4140 IF FF$ = "" THEN GOSUB 4210: VTAB 22: PRINT "SPEICHERT
     ";F$: IF F$ = "" THEN 2440
4150 GOSUB 4310
4160 PRINT D$;"BSAVE";F$",A$";FS$" ,L$";FL$
4170 GOSUB 4210: GOTO 2440
4200 REM ***************************************************
4210 POKE 35,23: VTAB 22: HTAB 1: CALL - 958: POKE 35,24:
     RETURN : REM 'Bitte warten' löschen
4300 REM ***************************************************
4310 VTAB 23: HTAB 1: CALL - 868: INVERSE : PRINT "  BITTE
     WARTEN  ";: NORMAL : VTAB 22: PRINT CHR$ (1): RETURN
4400 REM ***************************************************
4410 REM Hauptmenü
4420 TEXT : HOME : INVERSE
4430 HTAB HA: PRINT "ADRESSE";
4440 HTAB HC: PRINT "INHALT";: HTAB HS - 1: PRINT
     "BITMUSTER";
4450 HTAB HS + 11: PRINT "KOMMANDOS": NORMAL
4460 VTAB VS - 1: HTAB HS - 1: FOR I = 0 TO WI + 2: PRINT
     R$;: NEXT
4470 VTAB VS + LG: HTAB HS - 1: FOR I = 0 TO WI + 2: PRINT
     R$;: NEXT
```

```
4480 VTAB VS: FOR I = 1 TO LG: HTAB HS - 1: PRINT R$: NEXT
4490 VTAB VS: FOR I = 1 TO LG: HTAB HS + WI + 1: PRINT R$:
     NEXT
4500 REM ----------------------------------------------
4510 VTAB 3: RESTORE : FOR I = 1 TO DT: HTAB HS + 11: READ
     A$: PRINT A$: NEXT
4520 DATA (1 BUCHSTA-),BE EINGEBEN),,LADEN,SPEICHERN,#
     HEX-ADR,<-  ->,,LEERTASTE,I J K
     M,NEUSTART,<RETURN>,,WERTEAUSGABE,VERSCHIEBEN,↑INVERTIE
     REN,TEST 80 Z,,QUITTIEREN
4530 VTAB 24: HTAB 1: PRINT "FS$=";FS$;: HTAB 14: PRINT
     "FL$=";FL$;
4540 HTAB 26: PRINT "LG=";LG;: HTAB 35: PRINT "IV=";IV;
4550 VTAB VW: HTAB HW
4560 RETURN
4600 REM ************************************************
4610 REM Initialisierung
4620 WI = 7: REM Fensterbreite für Bitmuster
4630 DIM AR(WI + 2,15): REM Editier-Array
4640 DT = 19: REM Anzahl der DATA-Statements
4650 HA = 1:HC = 9:HS = 17: REM Startpositionen horizontal
4660 IV = 16:LG = 9: REM Intervall: Anzeigebereich
```

```
4670 R1$ = "5000":R2$ = "6FFF": REM Eingabebereich
4680 VS = 12 - INT (LG / 2): REM Startposition vertikal
4690 HW = 38:VW = 1: REM Cursor Warteposition
4700 FL$ = "1000": REM File-Länge
4710 D$ = CHR$ (4)
4720 R$ = "*": REM Rahmen des Bitmuster-Fensters
4730 CS$ = "+": REM Editier-Cursor
4740 FS$ = "5000": REM Ab hier wird File in Speicher
     geladen
4750 GOSUB 4920: REM HEX -> DEZ Umwandlung der Variablen
4760 GOSUB 4420: REM Menü
4770 GOSUB 2420: REM Kommandoeingabe
4780 VTAB 23: END
4900 REM ************************
4910 REM HEX -> DEZ Umwandlung der Variablen
4920 H$ = FS$: GOSUB 1220:FS = DEC
4930 H$ = FL$: GOSUB 1220:FL = DEC
4940 H$ = R1$: GOSUB 1220:R1 = DEC
4950 H$ = R2$: GOSUB 1220:R2 = DEC
4960 ST = FS: REM Startadresse
4970 RETURN
```

Nachtrag zum Pascal-RAM-Disk-Driver

Unsere Setzerei hat bei dem Beitrag RAM-Disk-Driver für Pascal 1.1, Peeker 6/85, Seite 48 einen Teil des Listings INIT.PASCAL.SOURCE weggeschnitten. Nach dem Macro PUSH fehlt die Macro-Definition LOAD. Es sind daher folgende Zeilen einzufügen:

```
    .MACRO LOAD
    LDY #00
$1  LDA %1,Y
    STA %2,Y
    INY
    CPY #%3
    BNE $1
    .ENDM
```

peeker Börse

Gelegenheitsanzeigen

Sie können unter dieser Rubrik zu einem besonders günstigen Preis

- Ihre Hardware und Software verkaufen
- Ihre Hard- und Software suchen
- Kontakte knüpfen und vieles mehr

Musteranzeige privat

1 Druckzeile à 32 Buchstaben nur DM 5,– zuzügl. ges. MwSt.

Beispiel:

> Verkaufe neuwertigen Typenraddrucker mit Apple-Interface. Preis auf Anfrage. Tel. 007

nur DM 17,10 inkl. MwSt.

Musteranzeige gewerblich

Die einspaltige Millimeterzeile (42 mm breit) kostet 5,50 zuzügl. ges. MwSt.

Beispiel:

> Neu im Angebot: Professionelle, separate Tastatur für Apple II plus 16 Funktionstasten und separatem Ziffernblock. Fa. Keyboard & Co.

nur DM 137,94 inkl. MwSt.

---

peeker MAGAZIN FÜR APPLE-COMPUTER

peeker-Börse

AUFTRAG FÜR KLEINANZEIGEN

Bitte veröffentlichen Sie in der nächsterreichbaren Ausgabe nachstehenden Text unter folgender Rubrik:

☐ suche Hardware ☐ suche Software ☐ Tausch ☐ Verschiedenes
☐ biete Hardware ☐ biete Software ☐ Kontakte ☐ Chiffre

Bitte jeweils 32 Buchstaben pro Zeile–einschließlich Satzzeichen und Wortzwischenräume. Bitte Absender nicht vergessen.
Chiffregebühr DM 6,– zuzügl. MwSt.

---

peeker MAGAZIN FÜR APPLE-COMPUTER

Produkt-Karte

Zu der in »**peeker**«, Heft_____, Seite_____ erschienenen
☐ Anzeige ☐ Produktbesprechung

über _____

_____

bitte ich um detaillierte Information.

Ich wünsche ☐ Prospekt, Datenblatt ☐ Preisliste ☐ schriftliches Angebot ☐ tel. Rückruf

Menge	Produkt und Bestellnummer	à DM	gesamt DM

gebe ich nebenstehende Bestellung unter Anerkennung Ihrer in der Anzeige genannten Liefer- und Zahlungsbedingungen auf.

_____
Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)

---

peeker MAGAZIN FÜR APPLE-COMPUTER

Info-Karte

Schreiben Sie uns, wenn Sie Fragen, Ergänzungen oder Anregungen zu Aufsätzen in »**peeker**« haben:

_____

_____

_____

_____

_____

_____

peeker-Börse

Vorname, Name

Beruf

Straße

Wohnort

PLZ

Bitte veröffentlichen Sie den umstehenden Text von _____ Zeilen à _____ DM in der nächsterreichbaren Ausgabe von **»peeker«**

Bei Angeboten: Ich bestätige, daß ich alle Rechte an den angebotenen Sachen besitze

Datum Unterschrift

Produkt-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

Anschrift der Firma angeben, bei der Sie bestellen bzw. von der Sie Informationen wünschen

Info-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

ANTWORTKARTE

peeker-Börse
Anzeigen-Service

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

POSTKARTE

Firma

Straße

PLZ/Ort

POSTKARTE

peeker
Redaktion

Postfach 10 28 69

6900 Heidelberg 1

Produkt-Karte

Wünschen Sie weitere Informationen zu einem der im Heft vorgestellten Produkte?

Nichts einfacher als das.
Produkt-Karte ausfüllen, mit 60-Pfennig frankieren und absenden.

Vorher aber nicht vergessen:
kreuzen Sie an, welchen Informationswunsch Sie haben.

Damit erleichtern Sie dem Hersteller eine gezielte Beantwortung Ihrer Anfrage

Zum Schluß tragen Sie auf der Rückseite die genaue Anschrift des Inserenten/Herstellers und Ihre vollständige Firmenanschrift ein.

PEEKER
MAGAZIN FÜR APPLE-COMPUTER

Peeker-Sammeldisketten

Einzelbezug DM 28,–
Fortsetzungsbezug DM 20,–
(Jederzeit kündbar, jedoch mindestens 6 Disketten)
(* = nur auf Diskette, nicht im Peeker gelistet! Seitenangaben beziehen sich auf Beginn des Listings)
Hüthig Software Service
Postfach 10 28 69 · 6900 Heidelberg 1

Disk# 1
(Heft 1+2, 1984)

T.DISASSEMBLER.65C02 (1/84, S. 15)
DISASSEMBLER.65C02

T.ACCEL.WAIT (1/84, S. 22)
ACCEL.WAIT
T.ACCEL.BOOT
ACCEL.BOOT
ACCEL.LC.KOPIERER
T.ACCEL.LC.KOPIE
ACCEL.LC.KOPIE
T.ACCEL.ROM.KOPIE1
ACCEL.ROM.KOPIE1
T.ACCEL.ROM.KOPIE2
ACCEL.ROM.KOPIE2

TURTLE.GRAFIK.MIT.REMS (1/84, S.29)
TURTLE.GRAFIK.OHNE.REMS *

DOUBLE.LORES.SOFTSWITCH.DEMO (1/84, S. 37)
DOUBLE.LORES.APPLESOFT.DEMO
AMPER.DOUBLE.LORES.DEMO
T.AMPER.DOUBLE.LORES
AMPER.DOUBLE.LORES
T.DOUBLE.LORES
DOUBLE.LORES

HIRES (1/84, S. 41)
T.PRINTHIRES
PRINTHIRES

DHGR.APSOFT.DEMO (2/84, S. 30)
AMPER.DOUBLE.HIRES.BAS
AMPER.DOUBLE.HIRES
T.AMPER.DOUBLE.HIRES
DHGR.LINEPLOTTER

INSTRING.TEST (2/84, S. 43)
INSTRING.OBJ
T.INSTRING.OBJ
INSTRING.LISA.SOURCE

LOESCHEN.EINES.ARRAYS (2/84, S. 52)

ULTRATERM.ENGLISCH * (2/84, S. 60)
ULTRATERM.DEUTSCH *

PRIMZAHLEN.OVERMEYER * (2/84, S. 70)
PRIM.OBJ0 *
PRIM.OBJ1 *
PRIM.TEST *
PRIM.TOOLKIT.SOURCE *

Disk #2
(Heft 1-2, 1985, DOS-Format)

T.RAMDISKLC (1-2/85, S. 14)
RAMDISKLC

T.IBS.RAMDISKDRIVER (1-2/85, S. 20)
IBS.RAMDISKDRIVER

T.AP20.RAMDISKTEST
AP20.RAMDISKTEST

T.QUICKCOPY (1-2/85, S. 26)
QUICKCOPY
QUICKCOPY.PUFFER
PRODOS.COPYA
T.PRODOS.COPYOBJ *
PRODOS.COPYOBJ

PRODOS.PATCH (1-2/85, S. 31)

T.APPLESOFT.FRE (1-2/85, S. 36)
T.LC.FRE
LC.FRE
FRE.TEST
T.RAM.FRE *
RAM.FRE

T.SCHIRMDISK (1-2/85, S. 44)
SCHIRMDISK.LISA.SOURCE
SCHIRMDISK

T.VIDEXT
VIDEXT.LISA.SOURCE
VIDEXT

GETPAS (1-2/85, S. 70)
T.GETPAS.ASS *
GETPAS.ASS
GETDOS.PASCAL.SOURCE
COPYDUPDIR.PASCAL.SOURCE

PRODOS.EDITOR.MACROS (1-2/85, S. 86)

Disk #3
(Heft 1-2, 1985, CP/M-Format)

STEUER.84 (1-2/85, S. 47)
PASS.BAS
MENUE.BAS
HELP.BAS *
A.BAS
B.BAS
C.BAS
D.BAS
E.BAS
F.BAS
G.BAS
H.BAS
I.BAS
J.BAS
K.BAS
L.BAS
M.BAS
N.BAS

Disk #4
(Heft 3 + 4, 1985)

TESTGENERATOR (3/85, S. 26)
SAETZE
BAHNFAHRT *
ZU *
TUN.UND.SOLLEN *
IRGEND *

MULTIPRECISION (3/85, S. 32)

T.WS.TRANSFER (3/85, S. 36)
WS.TRANSFER
T.WS.TRANSFER.2 *
WS.TRANSFER.2 *
GETCPM

PRIM.0.SC.SOURCE (3/85, S. 62)
PRIM.0.BIN

PRIM.1.SC.SOURCE
PRIM.1.BIN
PRIM.FP

ACCELERATOR.ABSTELLEN (3/85, S. 66)

T.WILDCARD.TEST * (3/85, S. 72)
WILDCARD.TEST1 *
T.WILDCARD.TEST2 *
WILDCARD.TEST2 *

XPLOT.DEMO (4/85, S. 18)
XPLOT.ROUTINE
T.XPLOT.ROUTINE

MENUE.GENERATOR (4/85, S. 22)

T.MACROS.65C02 (4/85, S. 31)

TERMINAL (4/85, S.36)
TERMINAL.B
T.TERMINAL.B

CAT.ARRAY (4/85, S. 44)
CAT.SAVER
EINTRAG.SUCHER
EINTRAG.ANALYSE
PRODOS.READER
T.PRODOS.READER.OBJ
PRODOS.READER.OBJ

MOUSESTUFF.PASCAL.SOURCE (4/85, S. 51)
MOUSE.ASS.PASCAL.SOURCE
TESTMOUSE.PASCAL.SOURCE
DRAWMOUSE.PASCAL.SOURCE

INALL.DATA (4/85, S. 70)
SCREEN80.DATA (4/85, S. 33)
SCREEN80.SAVER (4/85, S. 76)

Disk #5
(Heft 5, 1985, DOS-Format)

T.FM.BSP (5/85, S. 9)
FM.BSP

T.SLOTRAMDISK (5/85, S. 13)
SLOTRAMDISK
SLOTRAMDISK.HELLO

PLOT.2.0 (5/85, S. 20)
T.PLOT.B
PLOT.B
PLOT.PROTECTOR

T.CONVERT560 (5/85, S. 26)
CONVERT560
CONVERT560.DEMO

T.EDA (5/85, S. 33)
EDA

TRANSCEND.PASCAL.SOURCE (5/85, S. 36)

T.BLOCKTRACER (5/85, S. 51)
BLOCKTRACER
T.BLOCKTRACER1
BLOCKTRACER1

FORMAT.LC (5/85, S. 56)
FORMAT.LC.START

T.DISKDRIVER.DEMO
DISKDRIVER.DEMO

RANDOM.DEMO (5/85, S. 69)
COLUMN80.DEMO

SUPERDUMP.EPSON (6/85!)
SUPERDUMP.IMAGEWRITER
SUPERDUMP.BILD
T.SUPERDUMP
SUPERDUMP
EPSON
IMAGEWRITER

Disk #6

HELLO
ASMDIV

CURSOR1
T.CURSOR1
CURSOR2
T.CURSOR2
LINIE
T.LINIE
VIERECK
T.VIERECK
BOX
T.BOX
HINTERGRUND
T.HINTERGRUND
PAGE.SWAP
T.PAGE.SWAP

WANDERNDER.STRICH
KOMPRESSOR.DEMO
KREIS.1
KREIS.2
KREIS.3
FLIPPER
T.FLIPPER
KOMPRESSOR
T.KOMPRESSOR

OLYMPIA
T.OLYMPIA

FOURIER.MAIN
FOURIER.SYN
FOURIER.SPEC

AS.ERWEITERUNG
T.AS.ERWEITERUNG
AS.ERW.PROSTART
AS.ERW.PRO
T.AS.ERW.PRO

INSTALL.PASCAL.SOURCE
RAMDISK94.PASCAL.SOURCE
INIT.PASCAL.SOURCE

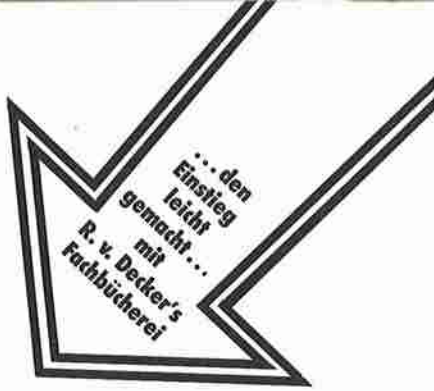
RAMDISK.INIT.DOS
AUXDRIVER
T.AUXDRIVER
MOVEDRIVER
T.MOVEDRIVER
RAMDISK.FORMATTER
T.RAMDISK.FORMATTER

SOLITAIRE.START
SOLITAIRE
SOLITAIRE.B
T.SOLITAIRE.B

Computer-Fibel

Von Karl Bolle 3., völlig neubearbeitete Auflage. 1985. 194 Seiten. Kartoniert. DM 26,–. ISBN 3-7685-6784-2

Welche organisatorischen Probleme gibt es bei der Computerinstallation am Arbeitsplatz? Was ist Hardware, Software, Orgware? Wie erstellt man ein Programm? Was ist Belegverarbeitung? Wie lauten die zehn Gebote für den Einstieg in die EDV? Diese und viele andere Fragen beantwortet die Computer-Fibel auf prägnante, unterhaltsame und anschauliche Weise. Sie versetzt den Leser in die Lage, sich innerhalb kürzester Zeit mit dem Computer-Basiswissen vertraut zu machen und die Grundzüge der Programmierung zu erlernen.



Von Martin Bergmann, Hans-Dieter Litke, Paul G. Maciejewski, Adelheid Kröz,

Dieses Buch enthält eine detaillierte Darstellung der Programmiersprache »C«. Es wendet sich an Leser, die über Grundkenntnisse der Informatik und über Programmiererfahrung verfügen, und ist damit für Hobbyinformatiker, Studenten naturwissenschaftlicher und technischer Fachrichtungen und für professionelle Software-Entwickler gleichermaßen geeignet.
Programmierneulingen wird empfohlen, sich zunächst mit einer einfacheren höheren Programmiersprache wie z. B. Pascal oder Basic zu beschäftigen.

Programmiersprache »C«

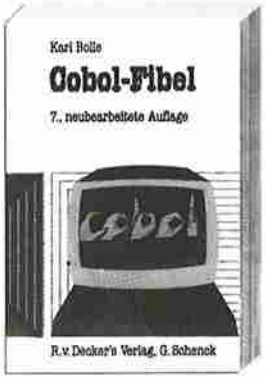
Esperanto der Software

Didaktisch-systematische Darstellung für den Anfänger und Anwender

1985. XII, 186 Seiten. Kartoniert. DM 38,–. ISBN 3-7685-6484-3

Cobol-Fibel

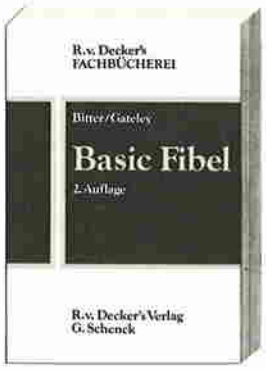
Von Karl Bolle 7., neubearbeitete Auflage. 1985. VIII, 267 Seiten. Kartoniert. DM 32,– ISBN 3-7685-6684-6



Bereits in 7. Auflage!

Autor dieses Buches ist ein erfahrener Programmierlehrer, der hier in einer allen Interessierten leicht zugänglichen Darstellung in ein umfangreiches, scheinbar kompliziertes Gebiet einführt. An kompletten Programmbeispielen wird die Systematik einer Programmiersprache deutlich, die für fast alle elektronischen Datenverarbeitungsanlagen gültig ist. Dem mit dem verstärkten Einsatz kleinerer Computer gegebenen Trend zur Bildschirmverarbeitung wird in dieser 7. Auflage durch zusätzliche Einfügung einiger Bildschirmprogramme Rechnung getragen. Ein bewährtes Werk, das sich ebenso zum Selbststudium wie als unterrichtsbegleitendes Material eignet.

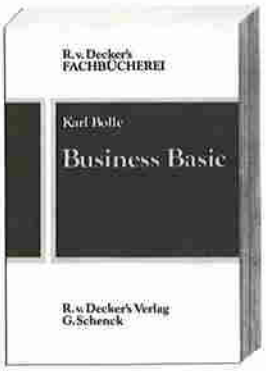
Diese leicht verständliche Einführung in die Programmiersprache BASIC eignet sich vorzüglich zum Selbststudium. Einige wenige Stunden genügen, um sich speziellen Problemen ohne Schwierigkeiten zuwenden zu können. Von BASIC gibt es viele Versionen, wenngleich diese Programmiersprache bei den unterschiedlichen Systemen schon einen weitgehend einheitlichen Stand aufweist. Die Grundlagen und gemeinsamen Sprachelemente von BASIC werden so vermittelt, daß der Lernende die Besonderheiten einer BASIC-Version ohne Probleme sich zu eigen machen kann.



Basic-Fibel

Von Professor Gary G. Bitter und Wilson Y. Gateley.
Aus dem Amerikanischen übersetzt und bearbeitet von Karl Bolle.

2. Auflage 1980. XIV, 154 Seiten. Kartoniert. DM 24,– ISBN 3-7685-2979-7



Business Basic

Von Karl Bolle.
1984. VIII, 223 Seiten. Kartoniert. DM 34,–
ISBN 3-7685-1283-5

Das vorliegende Buch ist für das Selbststudium von „Business BASIC" konzipiert worden. Business BASIC ist eine Mischung von FORTRAN, der verbreitetsten technisch-wissenschaftlichen Sprache und COBOL, der bekanntesten kommerziellen Programmiersprache. Business BASIC wurde speziell für die Dialogverarbeitung mit Mini-Computern und Büro-Computersystemen erfunden und weiterentwickelt. Der gesamte Sprachschatz kann mit Hilfe dieses Werkes, selbst von Anfängern, in etwa zwei Wochen erlernt werden.

Wordstar mit allen FX-80-Schriftarten

von Dipl.-Ing. H. A. Rohrbacher

Mit Hilfe einfacher Patch-Routinen wird gezeigt, wie man die von den Epson-Druckern FX-80 und FX-100 angebotenen Schriftarten Elite, Italic (= kursiv), Proportional und deren mögliche Mischungen für den Wordstar 3.0 anspricht und die Bildschirmdarstellung mit den deutschen Sonderzeichen erreicht.

Ohne Zweifel zählt Wordstar zu den weltweit bekanntesten Textverarbeitungs-Systemen, wobei die Wordstar-Version 3.0 gleichzeitig eine besonders weite Verbreitung erreicht hat und über eine Reihe willkommener Änderungsmöglichkeiten für eine alternative Druckersteuerung oder Bildschirmdarstellung verfügt. Wer hätte nicht gerne seinen Text durch Kursiv- oder Eliteschrift hervorgehoben, beides gemischt, oder die Proportionalschrift für eine optisch ausgeglichene Textdarstellung benutzt?

Die weit verbreitete Meinung, man brauche für abweichende Schriftarten jeweils eine individuell modifizierte WS-Version, ist irrig. Man kann sehr wohl die vom FX-80 angebotenen Schriftarten Pica (10 Zeichen/Zoll), Elite („Perl", 12 Zeichen/Zoll), Italic (= Kursiv) und Proportional (unterschiedliche Dickten = Buchstabenbreiten) in einer einzigen, einmal installierten WS-Version vereinen, miteinander mischen, dehnen, komprimieren, indizieren (= hoch- und tiefstellen), fett oder doppelt drucken, da der größte Teil dieser Möglichkeiten über ESC-Sequenzen aufrufbar ist. Die einzelnen WS-Befehle müssen daher lediglich den FX-80-ESC-Befehlen entsprechen.

Die normale PICA-Schrift kann in *Italic oder in* Elite umgewandelt werden. -> *auch eine Mischung ist möglich.* Wer's noch schmaler haben möchte, nimmt diese Eng-Schrift. Sperrschrift oder weit ist immer darstellbar. **Proportional-Schrift ist die Schönste!** Man kann auch Proportional-weit schreiben oder *Italic-weit*, eine gedehnte Elite sowie eine *weite Elite+Italic* zaubern. tief- oder hoch-gestellt und die Mixturen aus tief/eng oder hoch/eng sind bekannt. Doch: *tiefgestellte Elite+Italic* sind schon sehr hübsch! Natürlich kann die Sperrschrift indiziert werden. Breit hoch oder tief gefällt Grafikern. **Fett-Druck** bzw. **Doppel-Druck** ist in gleicher Zeile nur mit **Pica** mischbar. *Italic-Sperrschrift* hebt sich gut hervor *Doppeldruck von Italic/Elite ist am Zeilenanfang* problemlos. *Fettdruck dagegen nicht!* Generell **nicht mischbar** sind die vom **FX-80** nicht druckbaren Schriftkombinationen: Elite+eng, Elite+fett, Elite+Sperr, Propo.+eng und +Sperr, Italic+eng, Italic+Sperr und alle davon betroffenen **Indizierungen.** Das Übrige ist schön schön schön *schön* schön schön schön schön...

Bild 1

Die WS-Anpassung an diese Fähigkeiten wird durch ein „Patchen" bewirkt (wörtlich: „am Zeug flicken"), das auch von einem Ungeübten bei strikter Einhaltung der Vorgehensweise durchgeführt werden kann. Ziel der Bemühungen ist es also, eine Wordstar-Version zu erstellen, mit der beispielsweise ein in **Bild 1** gezeigter Misch-Text ohne Diskettenwechsel oder Collagetrick ausgedruckt werden kann.

Die Vorbereitung

Es wird davon ausgegangen, daß die Wordstar-Version 3.0 lauffähig, also auf den Apple II grundsätzlich angepaßt vorliegt. Weiterhin beziehen sich die folgenden Änderungen auf die Verwendung eines Epson-Druckers vom Typ **FX-80** oder **FX-100**. Mit einigen Erweiterungen wird auch der Druck-Computer LQ-1500 und mit Einschränkungen der Matrixdrucker RX-80 berücksichtigt.

Der erste Schritt beginnt damit, daß man sich eine WS-Kopie erstellt, wobei nur die drei Files **WS.COM**, **WSMSGS.OVR** und **WSOVLY1.OVR** vorhanden sein sollen. Alle drei gehören zusammen. Sollten beim Kopieren noch weitere Dateien übertragen worden sein, so werden diese gelöscht („ERA" unter CP/M oder „Y" unter WS). Nun wird die Befehlsdatei **INSTALL.COM**, am besten mittels „PIP.COM", zusätzlich auf die Diskette kopiert. Es schadet nie, von der so entstandenen Diskette wiederum eine Kopie herzustellen, mit der dann weitergearbeitet werden kann, wenn die Erstversion aus irgendeinem Grund defekt geworden sein sollte.

Das Installations-Programm

Die für den FX-80 vorzunehmenden „Patches" werden ausschließlich mit Hilfe des INSTALL.COM-Programms durchgeführt. Man beginnt damit, daß der auf der vorbereiteten Diskette enthaltene WS gestartet wird. Aus dem Haupt-Menue wählt man „R", wonach sich WS mit „COMMAND?" meldet.
Nun gibt man „INSTALL" (ohne „.COM") ein und startet damit das Installationsprogramm. Nach einigen Sekunden wird man gefragt:
– *Do you want a normal first time INSTALLation of WordStar?*
Da das sicher nicht der Fall ist (sonst wäre der WS bisher nicht gelaufen), antwortet man mit „N" für No.
Im nachfolgenden Menue werden die Antwortmöglichkeiten A, B, C und D angeboten; von diesen wählt man „D".
Danach folgt die Frage:
– *Filename of Wordstar to modify?*
Hier ist der Name der vorliegenden WS.COM-Datei, nämlich „WS" (ohne „.COM") einzugeben und mit Return abzuschließen.
Es folgen insgesamt acht weitere Fragen, die alle nacheinander mit Return beantwortet werden können, wenn – wie angenommen wird – eine 80-Zeichenkarte mit dem Videx-Standard und ein Epson-Interface (oder kompatibles) verwendet werden. Die 80-Zeichenkarte wird im Slot 3, die Interface-Karte für den Drucker im Slot 1 erwartet.
Nach dem achten Return fragt das Installationsprogramm:
– *Are the modifications to wordstar now complete? ok (Y/N):*
Da wir erst jetzt beginnen wollen, die „Patches" vorzunehmen, muß an dieser Stelle unbedingt mit „N" geantwortet werden. Als Folge davon ist man auf einer Arbeitsebene angekommen, die es erlaubt, die einzelnen Speicherzellen nach Wunsch zu ändern. Es wird angeboten:
– *Location to be changed (0 = End)*
Zum Aufruf des Speicherplatzes kann entweder die hexadezimale Adreßnummer eingegeben oder der Label-Name verwendet werden. Letzterer muß in jedem Fall immer mit einem Doppelpunkt (:) enden. Will man unterschiedliche WS-Versionen patchen, so empfiehlt es sich, die Label-Namen zu verwenden, da sich diese nie ändern, in einigen Fällen aber die relevanten Speicheradressen von Version zu Version andere Werte angenommen haben können. Wir bleiben beim WS 3.0 und müssen zunächst einige Unarten von WS bereinigen. Hierzu muß man wissen, daß bei jeder Installation zwei wichtige Dinge automatisch auf den US-Standard zurückgesetzt werden, die man hierzulande nicht sonderlich schätzt:

1. Die auf dem Bildschirm bisher bekannten deutschen Sonderzeichen, also die Umlaute, das ß und das §-Zeichen, erscheinen ohne Zutun wieder im ASCII-Format als {, }, [,], @ und \, und wer seinen Wordstar bisher nicht anders kannte, hat hier Gelegenheit zur Umstellung auf den deutschen Bildschirm. (Anm. der Red.: Diese „Unart" existiert nicht bei 80-Zeichenkarten mit Deutsch als erstem Zeichensatz.)
2. Der WS wird langatmig und führt große Warteschleifen zwischen den Ctrl-Kommandos und der Menue-Anzeige oder für die Anzeige des Ctrl-Zeichens ein, was die Geduld des flinken WS-Benutzers nur unnötig strapaziert.

An dieser Stelle sind daher vorab zwei Anmerkungen angebracht: Einmal werden deutsche Verhältnisse wiederhergestellt, indem im Label „TRMINI" = ab Speicherstelle 0292H das für die deutsche Zeichendarstellung wichtige Ctrl-Z3 untergebracht wird. Weiterhin sei auf einige Zeitschleifen hingewiesen, die ebenfalls bei jedem Patch-Vorgang neu festgelegt werden müssen, wenn man nicht zu lange warten möchte, bis der WS endlich zur Sache kommt. Die entsprechenden Labels sind mit DEL3: und DEL4: benannt und haben die Adressen 02D1H bzw. 02D2H.

^A = ESC₁		^N = ESC₂	
^AM	->	^NP	Elite
^A4	->	^N5	*Italic*
^A4^NM	->	^A5^NP	*Italic+Elite*
^Ap1	->	^Np0	Proportional
^W^Ap1	->	^Np0^Q	Propo weit
^E	->	^R	Engschrift
^W	->	^Q	weit
^W^E	->	^R^Q	Sperrschrift
^D^W	->	^Q^D	weit+dopp
^W^A4	->	^N5^Q	*Italic weit*
^D^W^A4	->	^N5^Q^D	*It.wt.dopp*
^T	->	^T	tiefgesetzt
^Y	->	^Y	hochgesetzt
^B	->	^B	fett
^D	->	^D	doppelt
^B^A4	->	^N5^B	*Italic fett*
^T^A4	->	^N5^T	*Italic tief*
^Y^A4	->	^N5^Y	*Italic hoch*
^T^AM	->	^NP^T	Elite tief
^Y^AM	->	^NP^Y	Elite hoch
^B^AM	->	^NP^B	Elite fett
^W^AM	->	^NP^Q	Elite weit
^T^E	->	^R^T	eng tiefgestellt
^Y^E	->	^R^Y	eng hochgestellt

kein ELITE+ENG, ELITE+FETT, ELITE+SPERR, ENG-PROPORTIONAL

Tabelle 1

Die mnemotechnische Ordnung

Wenn man sich den WS nach eigenem Geschmack optimal richten möchte, so ist das nur mit einem gut merkbaren Satz von Ctrl-Codes möglich. Leider können nur wenige Eselsbrücken aus dem Amerikanischen übernommen werden, und es ist geradezu grotesk, wenn zum Hochstellen des Textes ↑T und zum Tiefstellen ↑Y verwendet werden muß, wo doch jeder Deutschsprechende „T" als „Tief" interpretiert und mit „Y" einen nach oben gerichteten Vektor meint.
Da nicht viele Buchstaben für die verschiedenen Schriftarten zur Auswahl stehen, sollten diese wenigstens sinnentsprechend eingesetzt werden. WS-Benutzer, die sich an den US-Standard gewöhnt haben, sind nicht gleich zum Umdenken bereit. Aber spätestens nach ein paar Tagen folgt man gerne der Vereinfachung, und die verwirrenden, alten Ctrl-Befehle für den Aufruf der Sonderschriftarten sind schnell vergessen. Will man aber schon einmal geschriebene Texte nicht mehr ändern und beim gewohnten Ctrl-Satz bleiben, so genügt es, wenn für die neu aufzunehmenden Schriftarten lediglich zwei neue Ctrl-Codes definiert werden.
Die Festlegung der Codes ist eine Voraussetzung für das Patchen und muß daher zuvor erledigt werden. In der **Tabelle 1** sind die Befehle zur Auswahl der einzelnen Schriftarten zusammengefaßt, wobei eine ganze Reihe weiterer kombiniert werden könnten: ↑E steht für Engschrift, ↑W für Weit, ↑T für Tief, Rücksetzten von Eng wird ↑R usw. Die wichtigsten Ctrl-Codes sind ↑A und ↑N. Hier liegt der Schlüssel zum Ganzen:
Die im Amerikanischen oft für Fettschrift verwendeten Labels „PALT:" und „PSTD:" können freigemacht werden, da Fettschrift mittels ↑B aus dem Label „BLDSTR:" (= boldstrike) gesteuert werden kann. „PALT:" und „PSTD:" werden dafür zu offenen ESC-Sequenzen modifiziert und erlauben, zusammen mit dem für die typische Sonderschriftart angehängten Argument, die FX-80-Schriften aufzurufen und nach Wunsch zu mischen oder doppelt und fett zu schreiben. Mit ↑Ax = Alternative Schrift EIN bzw. mit ↑Px = Pica = Alternative Schrift AUS schaltet man die Sonderschriften ein und aus. Das Argument „x" steht hierbei für die einzelnen Schriftarten wie z.B. Elite, Italic und Proportional.

Die zu ändernden Zellen

Da auf dem Bildschirm noch immer die Eingabe der „Location to be changed" erwartet wird, beginnen wir mit den zuvor erwähnten Sonder-Patches für die deutsche Darstellung und den kurzen Warteschleifen.
Die Vorgehensweise ist folgende: Wir wählen die Adreßnummern und nicht die Label-Namen, da wir innerhalb der Version 3.0 bleiben, und geben als erstes „292 <R>" ein, wobei <R> für Return steht. Auf dem Bildschirm wird uns dann der alte Inhalt der Speicherzelle 0292H, nämlich „00" angezeigt. Beim Anwählen der Adresse 0292H wurden die nichtführende Null und das „H" (für hexadezimal) weggelassen.
Der Inhalt einer Adresse wird geändert, indem man das für die Änderung benötigte Byte (wiederum hexadezimal) eingibt: 04 <R>. Wir haben mit 0292H das Label „TRMINI:" angewählt, das für die TERMINAL INITIALIZATION verantwortlich ist und von dessen acht Bytes vier belegt werden müssen, um Ctrl-Z3 für die deutsche Zeichendarstellung auf dem Bildschirm „einbauen" zu können. Mit „04 <R>" wird das „Zählbyte" für die noch folgenden vier Bytes vorangestellt, das festlegt, wieviel Bytes insgesamt im jeweiligen Label berücksichtigt werden und das beim Patchen zur Vollständigkeitskontrolle herangezogen werden kann. Jetzt erst folgen die vier weiteren Bytes: 14 in 0293H, 48 in 0294H, 1A in 0295H und 33 in 0296H.
Es wäre nun sehr mühsam, wenn jedesmal die Folgeadresse neu eingegeben werden müßte. Ein zweites <R> führt unmittelbar zur nächsten Adresse und der Angabe deren Inhalts, so daß die insgesamt fünf Bytes ab Adresse 0292H auch so eingegeben werden können:
292 <R> 04 <R><R> 14 <R><R> 48 <R><R> 1A <R><R> 33 <R>. Der Byte-String wird am Ende nur mit einem <R> abgeschlossen. Damit können gleich weitere, neue Adressen angewählt werden. Hierzu noch ein Beispiel für die Verkürzung der schon genannten Warteschleife bei Adresse 02D1H:
Auf „Location to be changed?" antwortet man: 2D1 <R> und bekommt den derzeitigen Speicherinhalt „19" ausgewiesen. Der Wert für die Warteschleife kann zwischen 1 und 127 liegen. Wir wählen den kürzesten und geben 01 <R> ein. Die nichtführende Null könnte man hier ebenfalls weglassen, es hätte also 1 <R> genügt, doch beim erstmaligen Patchen führt allzu große Freizügigkeit leicht zu Fehlern, weshalb es ratsam ist, die Bytes immer ungekürzt einzugeben.

Adresse:	Adr.#	CtrlP	Inhalte	Bedeutung
TRMINI:	Ø292		Ø4 14 48 1A 33	CTRL-Z3 -> deutsche Zeichen
DEL3:	Ø2D1		Ø1 (ex: 19)	mittellange Warteschleife AUS
DEL4:	Ø2D2		Ø1 (ex: 4Ø)	lange Warteschleife AUS
DEL5:	Ø2D3		ØØ (ex: Ø9)	Ctrl-delay AUS
PALT:	Ø6B5:	↑A	Ø1 1B	ESC-Sequ.-> Sonderschrift EIN
PSTD:	Ø6BA:	↑N	Ø1 1B	ESC-Sequ.-> Sonderschrift AUS
USR1:	Ø6C9:	↑Q	Ø3 1B 57 ØØ	ESC W Ø -> Weit AUS
USR2:	Ø6CE:	↑W	Ø3 1B 57 Ø1	ESC W 1 -> Weit EIN
USR3:	Ø6D3:	↑E	Ø1 ØF	SI=CHR$(15) -> Eng EIN
USR4:	Ø6D8	↑R	Ø3 12 1B 5Ø	DC + ESC P -> Eng AUS+NORMAL
RIBBON:	Ø6DD	↑Y	Ø3 1B 53 ØØ	ESC S Ø -> Hochstellen EIN
RIBOFF:	Ø6E2	↑Y	Ø4 1B 54 1B 48	ESC T +H -> Hochstellen AUS
ROLUP:	Ø6BF	↑T	Ø3 1B 53 Ø1	ESC S 1 -> Tiefstellen EIN
ROLDOW:	Ø6C4	↑T	Ø4 1B 54 1B 48	ESC T +H -> Tiefstellen AUS
PSINIT:	Ø6E7		Ø7 1B 4Ø 1B 52 Ø1 1B 4F	-> Drucker INIT
PSFINI:	Ø6F8		Ø6 1B 57 ØØ 12 1B 4Ø	-> Drucker END
BLDSTR:	Ø691	↑B	Ø3	Fettdruck 3-mal Anschlag
DBLSTR:	Ø692	↑D	Ø2	Doppeldruck 2-mal Anschlag
INITPF:+1	Ø367		48	(72 Zeilen Papierformat = .PL72)
INITPF:+2	Ø368		4Ø Ø2	(Papierlänge in 1/48" wiederholen)
INITPF:+5	Ø36B		ØØ	(entspricht .MTØ = Ø zusätzl. Kopfzeilen)
INITPF:+6	Ø36C		ØØ ØØ	(wird hier in 1/48" wiederholt)
INITPF:+13	Ø373		ØA	(entspricht .MBlØ = lØ Zeilen Fußabstand)
INITPF:+14	Ø374		5Ø ØØ	(wird hier in 1/48" wiederholt)
INITPF:+17	Ø377		Ø1	(Abstand Seiten-# vom Text in Zeilen = 1)
INITPF:+18	Ø378		Ø8 ØØ	(wird hier in 1/48" wiederholt)
INITPF:+24	Ø37E		Ø8	(Heftrand = 8 Spalten, entspricht .PO8)
INITRM:	Ø38Ø		43	(rechter Rand + 1 Spalte, entspr. ↑OR68)

Patches für den mnemotechnichen WORDSTAR 3.Ø mit den frei wählbaren, zusätzlichen Sonderschriften Elite, Italic und Proportional.

Tabelle 2

Patch-Tabelle

In der **Tabelle 2** sind alle für den mnemotechnischen Wordstar nötigen Speicheradressen und deren Inhalte zum Aufruf der FX-80-Schriftarten: Eng, Breit, Sperrdruck, Index, Elite, Italic, Proportional, deren mögliche Mischungen sowie der Fett- und Doppeldruck aufgeführt und die Patches für weitere, wichtige Bereiche der Druckdarstellung angegeben.
Die Reihenfolge der Vorgehensweise ist beliebig. Am besten beginnt man mit den Adressen 0292H, 02D1H und 02D2H, die beim Installieren immer neu gepatcht werden müssen.

Patch-Arbeit abschließen

Wenn alle Adresseninhalte gesetzt sind, wird die Arbeit einfach dadurch abgeschlossen, daß auf die Frage „Locations to be changed (0 = End)" eine Null mit <R> eingegeben wird.
Der sich anschließende Aufruf „Confirm terminal and printer selections ok (Y/N):" wird mit „Y" beantwortet. Danach läuft das Laufwerk an und kopiert den modifizierten, gepatchten WORDSTAR WS.COM auf die Diskette. Sobald dieser Vorgang beendet ist, wird WS automatisch neu aufgerufen, wobei die langen Warteschleifen fehlen, und man ist rasch auf der Haupt-Menue-Ebene angelangt. Es reizt nun zu sehen, ob alle Patches gelungen sind. Dazu legt man eine neue Test-Datei mit einem Umlaut im Namen an. Der Dateiname sei „HÜTHIG.TST". Zunächst schreibt man ein paar belanglose Worte wie: „...das ist Pica, dann: ↑AM ...das ist Elite... ↑NP ...wieder normal Pica... ↑A4 ...das ist Italic... ↑N5 ...wieder normal Pica... ↑Ap1 ...Proportional... ↑Np0 und wieder normal". (↑ bedeutet hier Ctrl-P und muß stets allen Befehlen vorangesetzt werden).
Nach erfolgtem Abspeichern dieses kurzen Textes und anschließendem Ausdrukken zeigt sich dann, ob das Werk den Erwartungen entspricht.

Tastatur und Ctrl-Codes

Eine Reihe von Apple-Nachbauten verfügt über frei programmierbare Funktionstasten mit einer Batteriepufferung. Der Wert solcher Einrichtungen ist unschätzbar, da lediglich eine vorprogrammierte Taste zum Einschalten einer Schriftart bzw. eine zweite zum Ausschalten gedrückt werden muß. Moderne Tastaturen, die an den Apple angeschlossen werden können, haben teilweise 10 bis 24 Funktionstasten, die auf bis zu vier Ebenen belegbar sind und

meist schon in einer Ebene über die wichtigsten Wordstar-Ctrl-Codes verfügen.

Das Pitch-Problem
Da die verschiedenen Epson-Schriftarten unterschiedliche Zeichen-Dichten aufweisen, ist der rechtsbündige Blocksatz nicht mehr durchführbar. Lediglich die Normalschrift Pica und Italic und deren Indexschriften haben den gleichen Pitch mit 10 Zeichen pro Zoll. Elite hingegen schreibt mit 12 Z/Zoll, Engschrift (condensed) mit 17 Z/Zoll, und die Sperrschrift ist 8,5 Z/Zoll breit. Die Breitschrift selbst verfügt über einen 5er-Pitch, und jede Mischung führt zu neuen Pitch-Werten. Schließlich gibt es für Proportional überhaupt keinen festgelegten Pitch, da die einzelnen Zeichen zwischen 5 und 12 Pixelpunkte breit sein können.

Man kann im Regelfall davon ausgehen, daß jede Datei in einer Vorzugs-Schriftart angelegt wird, wobei für Hervorhebungen, einzelne Passagen, Überschriften oder Teile von Tabellenwerken Sonderschriften benutzt werden.
Für den Ausdruck des Textes wird im allgemeinen ein linker Heftrand mit acht Spalten Abstand gewählt, der in der Adresse 037EH fest (default) installiert ist oder durch den Punktbefehl .PO (Druckspalte) frei gewählt werden kann. Somit bedeutet Linksbündigkeit = 8 x 1/10 Zoll = 20,32 mm Abstand vom linken Papierrand bei Normalschrift. Begänne jedoch der Text einer neuen Zeile nicht im Normalschriftmodus, sondern mit Engschrift, so wären nur 8 x 1/17 Zoll = 11.95 mm Heftrand festgelegt, und der eng geschriebene Text würde nach links vorgerückt erscheinen. Das Problem der Linksbündigkeit läßt sich prinzipiell vermeiden, wenn überhaupt kein Heftrand (.PO=0 bzw. Speicherinhalt von 037EH: 00) festgelegt ist.
Eine weitere Lösung besteht darin, daß in jedem Fall eine neue Zeile mit Pica-Normalschrift begonnen wird. Dies wiederum ist in der Praxis nicht akzeptabel. So helfen entweder nur eine bestimmte Anzahl von Leerzeichen je Schrifttyp zu Beginn der Zeile oder ein entsprechender Punktbefehl weiter. Beide Möglichkeiten sind in **Tabelle 3** zusammengefaßt und sind hinsichtlich der wählbaren Befehlsform gleichwertig. Punktbefehle müssen, wenn Normalschrift am Zeilenanfang wiederkehrt, wieder auf den Standardwert zurückgestellt werden; in unserem Fall also mittels .PO8 zu Beginn der neuen Normalzeile.

.po	Schrift
.po 6	**Proportional** a) entweder fortlaufend schreiben und zu Beginn der neuen Zeile = 2 Blancs setzen, oder: b) neue Zeile mit .po6 in Zeile zuvor richten
.po10	Elite a) entweder fortlaufend schreiben und zu Beginn der neuen Zeile = 4 Blancs geben, oder: b) neue Zeile mit .po10 in Zeile davor richten
.po 8	*Italic* a) entweder fortlaufend schreiben und zu Beginn der neuen Zeile = 2 Blancs geben, oder: b) neue Zeile mit .po8 in Zeile davor richten
.po13	Engschrift a) entweder fortlaufend schreiben und zu Beginn der neuen Zeile = 6 Blancs setzten, oder: b) neue Zeile mit .po13 in Zeile davor richten
.po 7	Sperrschrift a) entweder fortlaufend schreiben und zu Beginn der neuen Zeile = keine (!) Blanc-Lösung, nur: b) neue Zeile mit .po7 in Extrazeile davor richten
.PO4	Breit a) keine Blanc-Lösung, da BREIT über Li-Rand, nur: b) neue Zeile mit .po4 in Extrazeile davor richten

Alle Werte gelten nur für den DEFAULT- .PO8 - Wert von WS !

Tabelle 3

Individuelles Druckformat
Hat man sich schon einmal der Patch-Arbeit angenommen, so besteht die Gelegenheit, gleichzeitig und ohne Aufwand die Formatierung des Druckbildes nach eigenen Wünschen neu zu gestalten. Diese Vorzugseinstellung (default) wird dann bei Beginn der Arbeit automatisch vom Wordstar gewählt. Es ist ärgerlich, wenn bei jeder neu anzulegenden Datei erst einmal .PL72 gesetzt werden muß, nur weil der im Original vorliegende amerikanische WS ein .PL66 für das in den USA gebräuchliche und dort nur 11 Zoll lange Papier einstellt, während wir in Europa 12-Zoll-Papierbögen verwenden. Auch schätzt nicht jeder die drei zusätzlichen Leerzeilen am Kopfrand bei gleichzeitig nur sechs freien Fußzeilen, wodurch Punktbefehle wie .MT0 und .MB10 vorangestellt werden müssen. Ein gut gepatchter WS benötigt überhaupt keine Punktbefehle für die Standard-Druckmaske. Jedermann hat aber die Möglichkeit, seinen indivudellen Vorstellungen mit Hilfe des INSTALL-Programms freien Lauf zu lassen. Patch as patch can: Individuell und universell zugleich, das ist die besondere Stärke von Wordstar!

(Der zweite Teil dieses Aufsatzes unter dem Titel „Wordstar druckt internationale Zeichensätze" erscheint im nächsten Peeker.)

Blick über den Zaun:
Die moderne Satztechnik
Ein Mikrocomputer kann in Verbindung mit einem Matrixdrucker die Fähigkeiten einer Satzanlage bestenfalls nachahmen, aber niemals erreichen. Werfen wir hierzu einen Blick über den Zaun in das grafische Gewerbe:
Die heutigen Druckschrift-Zeichen haben in der Regel eine Dickte (= schriftzeichenabhängige Breite = character width), die x/18tel des breitesten Buchstabens des jeweiligen Schriftgrades einnimmt. Ein „v" nimmt beispielsweise 9/18 des großen „W" mit 18/18 ein. Ein 8-Punkt-„v" (1 Punkt = 0,375 mm) wäre damit effektiv 9/18 * 8 * 0.375 = 1,5 mm breit. Nun gibt es jedoch von *derselben* Schriftart, z.B. Helvetica, unterschiedlich *breit laufende* Schriftschnitte, z.B. Helvetica schmalmager, breithalbfett usw. Während bei einem normalen Schnitt der breiteste Buchstabe „W" ein „Geviert" (= gedachtes Buchstabenquadrat) einnimmt, belegt bei-

spielsweise beim schmalmageren Schnitt der breiteste Buchstabe nur ein „gestauchtes" Rechteck. Dieser Sachverhalt wird als „Set" bezeichnet (= schriftschnittabhängige Breite). Der Begriff „Pitch" (von Matrixdrucker-Herstellern geprägt?) entspricht annähernd dem Begriff „Dickte" bzw. „Set". Was die Pixel oder Nadeldrucker-Punkte anbelangt, so wird im Kathodenstrahl-Lichtsatz ein einzelner Buchstabe meist durch eine 128-mal-128-Bildpunkt-Matrix aufgelöst. Bei einer 8-Punkt-Schrift würde dies bedeuten, daß auf einem „Geviert" von 3 mal 3 mm maximal über 160.000 Bildpunkte belichtet werden könnten. Es ist somit klar, daß man selbst mit einer Lupe bei Druckbuchstaben keine „Pünktchen" mehr erkennen kann. In der Reprografie würde dies einem 400er „Raster" entsprechen. Unter Raster versteht man in Deutschland die Anzahl der Punkte oder Punktlinien pro Zentimeter, in den USA pro Zoll (dpi = dots per inch; Zoll ca. 2,5cm).

Beispielsweise beträgt die Punktauflösung beim Macintosh oder beim Apple IIe mit Double Hires ca. 80 dpi; dies entspricht etwa einem 32er Raster, wie man ihn bei Tageszeitungen verwendet. Zum Vergleich haben hochwertige Farbreproduktionen einen 70er bis 80er Raster. Aber die Druckschrift selbst bringt es dank Lichtsatz auf über 1000 dpi = über 400er Raster. Würde der Macintosh oder Apple IIe über eine Bit-Map-Grafik mit solcher Hochauflösung verfügen, so wäre ein Zusatz-RAM von umgerechnet ca. 7 Megabytes erforderlich. Weder ein 6502 noch ein 68000 wäre jedoch in der Lage, diese „Pixel-Masse" zu bewältigen.

Die Firma Compugraphic in Langen teilte mir hierzu mit, daß beispielsweise die cg 8600 sogar mit einer Auflösung von 2048 Linien pro cm arbeitet, wobei allerdings keine Bildpunkte, sondern Vektoren gespeichert werden. Der Peeker wird auf einer Linotype mit 300 Linien pro cm gesetzt (Tageszeitungssatz!). *U. Stiehl*

ProDOS-Editor 1.0

Applesoft-Editor unter ProDOS-Betriebssystem

von U. Stiehl

1984, Diskette und Manual, DM 98,–
ISBN 3-7785-1024-X

Mit diesem neuen Editor – übrigens der bislang einzige deutsche ProDOS-Editor – wird dem Applesoft-Programmierer ein Werkzeug zur effektiven Programmierung unter dem Betriebssystem ProDOS gegeben, denn die früheren Editoren sind allesamt unter ProDOS nicht mehr lauffähig.

Unter anderem sind folgende Features implementiert worden:

- Zeilenorientierter Editor mit jedem erdenklichen Redigierkomfort (Insert, Delete, Tab, Restore, freie Cursorbewegung in allen vier Richtungen, Eingabe von Ctrl-Buchstaben in Applesoft-Zeilen usw.)
- Renumber (Zeilen-Umnumerierung)
- Xreference (sortierte Variablenliste)
- Suchen von Tokens, Strings und Variablen
- dezimale und hexadezimale Umrechnungen
- Ausführung von Monitorbefehlen aus dem Editor heraus
- Listen des Applesoft-Programms in speicherinterner Form als Hex-Dump
- Suchen von Hex-Folgen, Adressen oder Speicherstellen im gesamten RAM-Bereich einschließlich der Language-Card
- frei definierbare Tastatur-Macrobefehle

Der Applesoft-Editor liegt in einem von ProDOS geschützten Bereich und läßt sich per Tastendruck vorübergehend abschalten und ebenso einfach wieder aktivieren.

Gerätevoraussetzung: Apple II+, IIe oder IIc

Hüthig Software Service,
Postfach 10 28 69,
D-6900 Heidelberg

r+r electronic
Elektronik • Fachliteratur • Personal-Computer

DataMagnetics-Disketten SS,SD 10 Stk 29.90

BASF qualimetric®
Disketten 1.Wahl
BASF-Disketten softsektoriert in Pappkarton
5.25"-Disketten alle **ohne Aufpreis** mit Verstärkungsring

Type	Beschreibung	10 Stk
1XV	**48 TPI, 1-seitig, einfache Dichte**	**39,90**
1DV	48 TPI, 1-seitig, doppelte Dichte	49,50
2/96V	96 TPI, 2-seitig, doppelte Dichte	82,50

3M Sicherheits-Disketten
5.25"-Disketten **mit Verstärkungsring**; softsektoriert

744-0	48 TPI, 1-seitig, doppelte Dichte	49,50
745-0	48 TPI, 2-seitig, doppelte Dichte	72,50
747-0	96 TPI, 2-seitig, doppelte Dichte	89,50

Disketten – Doubler nur noch .. 12,50
Zum Anbringen der zusätzlichen Schreibkerbe

5,25" – Plastikbox UNSER HIT! 5.95
Formschöne Klappbox aus Hart-PVC für ca. 10 Disketten in den Farben grün, orange, schwarz, hellgrau, elfenbein oder anthrazit. Bitte die gewünschte Farbe angeben. **Mischbar. Staffelpreise: 1 St. 5.95 10 St. 49.50 100 St. 459.–**

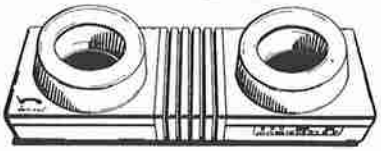
5,25" – Disketten-Kästen
Diskettenbox ohne Schloß 24,95
Repräsentative Disketten-Box aus deutscher Fertigung in den Abmessungen (T x B x H) 306x174x144 mm ohne Schloß mit topas farbenem Deckel für **ca. 80 Disketten** und 4 beschriftbaren Stützplatten. **Sehr preiswert!**

Diskettenbox 29.50
aufklappbar
für ca. 50 Disketten
aus stoßfestem, antistatischem transparentem Acryl mit 4 beweglichen, beschriftbaren Stützplatten.

Diskettenbox **abschließbar** 39.50
für ca. 85-100 Disketten
aus antistatischem ABS-Kunststoff, Deckel aus hochwertigem Acryl (rauchglasfarben). Der Trog kannplatzsparend in den Deckel gestellt werden. 9 beweglichen Unterteiler mit beschriftbaren Stützplatten sorgen für Übersicht.

Akustikkoppler s21d 289.–
300 Baud Modem CCITT V.21 Standard

Akustikkoppler s21d 289.–
300 Baud Modem CCITT V.21 Standard

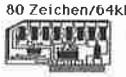


*** Mit FTZ-Nummer * Gebühren- u. anmeldefrei * Anschluß an alle Computer mit V24-Schnittstelle * Vollduplexbetrieb * Answer- und Originate-Modus * MADE IN GERMANY ***

TELETERM 2+/e 198.–
Telekommunikationssoftware für den Apple * ohne serielles Interface * Datenübertragung über den Gameport * Anschlußfertig an s21d

Unser großer Hit auch für Ihren Apple //e

80 Zeichen/64kB-RAM für APPLE //e



dtsch. Qualitätsprodukt incl. Demo-Diskette und ausführlichem Handbuch.

jetzt nur noch..... 195.–

8 Bit/16 Kanal A/D-Wandler 490.–
Preisgünstiger universeller A/D-Wandler. Kurzdaten: 8-Bit-Auflösung, R_e = 1 MOhm, 15 usec/Messung.

Accelerator //e 1398.–
Mit dieser Karte laufen Ihre Programme bis zu 3,5 mal schneller. (NCR 65C02; 3,5 MHz Takt, 80 kB-RAM dyn.)

Erphi-Controller AFDC-2 ..... 287.–
Dazu passende Laufwerke 80 Track (TEAC FD-55F) auf Anfrage.

Software-Hits für den Apple //e:

Print Shop die perfekte Heimdruckerei 139.50
mit der man in wenigen Minuten Grußkarten, Visitenkarten und sogar Banner entwerfen und ausdrucken kann.

Dazzle Draw das hochwertige Grafik-Set 179.–
Hochwertiges Grafik-Set für die verschiedensten Anwendungen. Ob für Computerkunst oder Bauzeichnungen.

Hardware-Hits für den Macintosh:

ThunderScan 1198.–
Lesekopf für Imagewriter (anstelle des Farbbandes) zum übertragen von Grafiken oder Text in MacPaint-Dokumente.

Die Aufrüstung Ihres 128k MAC auf 512k kostet bei uns nur: 1290.–

Apple//e, //c + Macintosh Vorführ- und Messegeräte preisgünstig abzugeben.

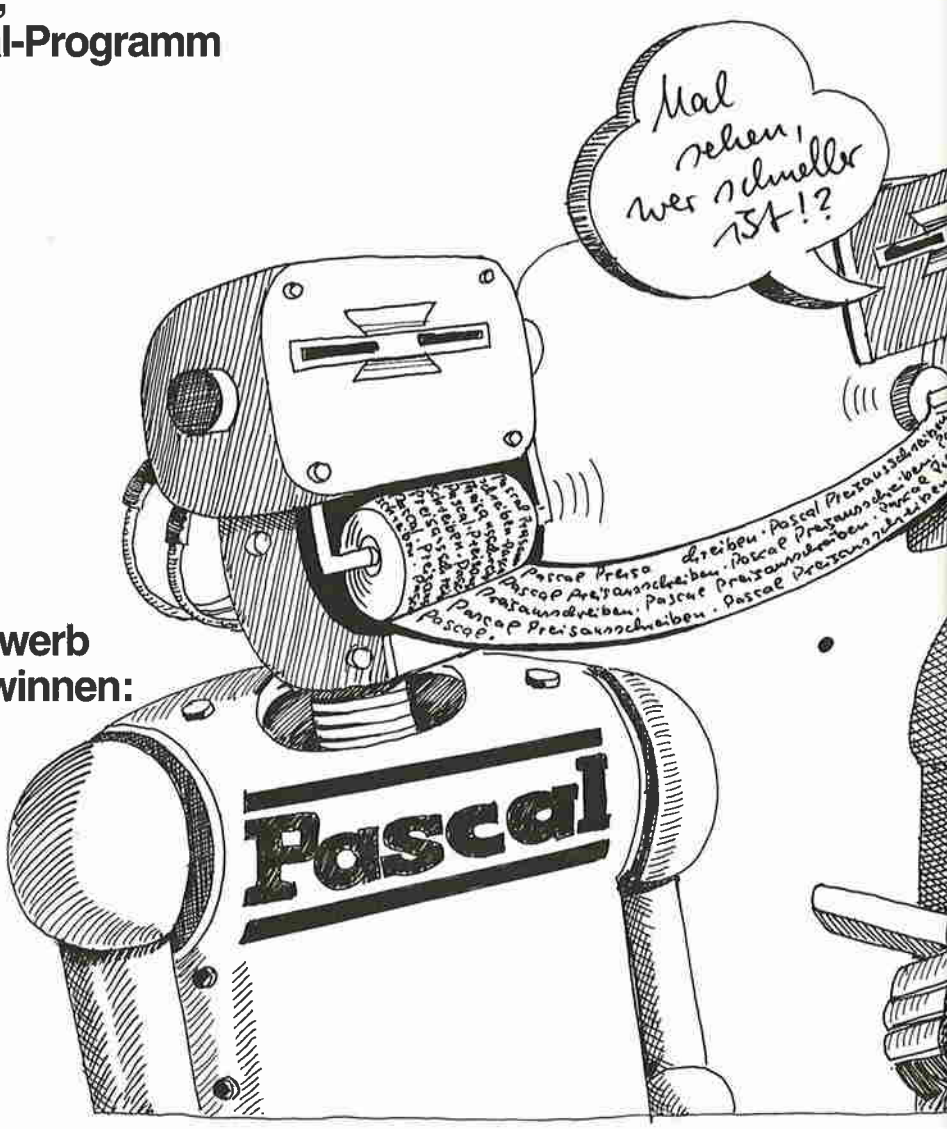
Ladenverkauf, Vorführung und fachgerechte Beratung

r+r electronic
Elektronik · Fachliteratur · Personal-Computer
6900 Heidelberg 1 Breslauerstr. 29 Tel.06221/781500
Geschäftszeiten: Mo. - Fr. 9-13 +14 - 18 Sa. 9 -13 Uhr
Versandanschrift: 6900 Heidelberg 1 Dammweg 2

Pascal-Preisausschre

Da heißt es mitmachen!

Es wird die Aufgabe gestellt, mit einem „getunten" Pascal-Programm einen ProDOS-TXT-File in einen Pascal-TEXT-File zu konvertieren.

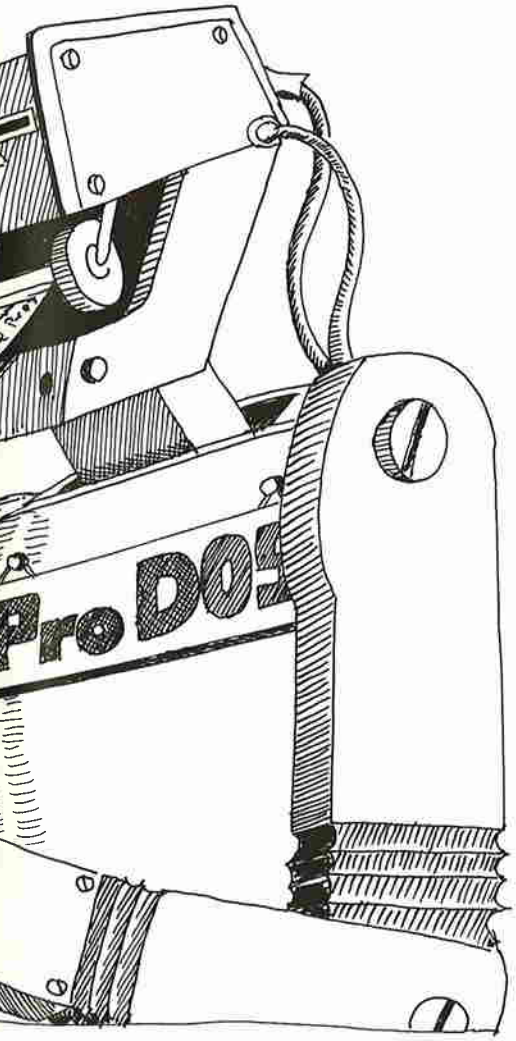


Bei unserem Pascal-Wettbewerb können Sie diese Preise gewinnen:

1. Preis: DM 500,-
2. Preis: DM 250,-
3.–25. Preis: Buchtitel

PREISAUSSCHREIBEN
Pascal Wettbewerb

iben



Sie erinnern sich sicherlich noch an das erste Peeker-Heft (September 1984), das ein Primzahlen-Preisausschreiben enthielt. Aufgabenstellung: Errechne die Primzahlen im Bereich 2–8191 so schnell wie möglich. Damals gingen 279 Lösungen – mit der sensationellen Bestzeit von 0,05s – ein, die allesamt in Assembler geschrieben waren. Bei der Durchsicht der Lösungen ist mir dann „gedämmert", daß ich bei dem Primzahlen-Wettbewerb nicht nur auf Geschwindigkeit, sondern auch auf Kompaktheit des Programms hätte abheben sollen. Damit hätten sich unelegante Lösungen mit zum Teil 8K Objektcode verhindern lassen. Deshalb wird unser neues Preisausschreiben – diesmal für Pascal-Fans – beide Aspekte berücksichtigen.

Die Übertragung und Konvertierung von Pascal-TEXT-Files auf ProDOS-Disketten oder von ProDOS-TXT-Files auf Pascal-Disketten hat im Gegensatz zur Errechnung von Primzahlen einen praktischen Nutzwert: Pascal-Programmierer können nämlich dann den Quellcode mit Appleworks, das unter ProDOS läuft, sehr komfortabel redigieren. Hierzu sind zwei verschiedene Utilities erforderlich:

1. PASTOPRO (From Pascal to ProDOS): Dieses Hilfsprogramm konvertiert Pascal-Textfiles in ProDOS-Textfiles.
2. PROTOPAS (From ProDOS to Pascal): Dieses Hilfsprogramm konvertiert ProDOS-Textfiles in Pascal-Textfiles.

1. PASTOPRO

Gegenstand des Pascal-Preisausschreibens ist die PROTOPAS-Utility. Da ich zur Erläuterung der Wettbewerbsbedingungen aus verständlichen Gründen kein Muster der PROTOPAS-Utility vorstellen kann, werde ich statt dessen eine PASTOPRO-Utility beschreiben (siehe Listing). Diese leistet im einzelnen folgendes:

1. PASTOPRO wird unter dem ProDOS-Betriebssystem mit RUN PASTOPRO von Slot 6, Drive 1 gestartet.
2. Danach erscheint ein spartanisches Menü, das Sie auffordert, eine Pascal-Diskette in Slot 6, Drive 2 einzulesen.
3. Nach „W = WEITER" werden am Bildschirm alle TEXT-Files, die sich auf der Pascal-Diskette befinden, angezeigt. Es werden *nur* TEXT-Files (also keine CODE-Files usw.) ausgewiesen, denen eine fortlaufende Nummer von 1 bis N vorangestellt ist, z.B. „1 SYSTEM.WRK.TEXT, 2 TEMP.TEXT" usw.). Nun wird man aufgefordert, die gewünschte Datei-Nummer einzugeben.
4. Gibt man keine Datei-Nummer im Bereich 1 bis N oder lediglich Return ein, so wird das selektive Pascal-Directory erneut angezeigt. (Man kann vorher die Pascal-Diskette wechseln.)
5. Gibt man eine existente Datei-Nummer ein, so beginnt der Übertragungsvorgang von der Pascal- auf die ProDOS-Diskette mit einem Piepston und der Anzeige von „*** START". Der Übertragungsvorgang bzw. das Programm selbst wird mit einem erneuten Piepston und der Anzeige von „*** ENDE" beendet. Die Zeitdifferenz zwischen erstem und zweitem Piepston ist die (von uns handgestoppte) Kopierzeit.

2. PROTOPAS

Das Wettbewerbsprogramm PROTOPAS muß folgende Voraussetzungen erfüllen:

1. PROTOPAS.TEXT muß als einheitlicher, d.h. nicht aufgeteilter, Quelltext (SYSTEM.WRK.TEXT = PROTOPAS.TEXT) vorliegen, der unter Pascal 1.1 **und** 1.2 „normal" compiliert werden kann, d.h. ohne Assembler, Linker usw. Zu diesem Zweck sollte also die Diskette neben SYSTEM.WRK.TEXT höchstens folgendes *Standard*-Files von Pascal 1.1 oder 1.2 enthalten:
SYSTEM.APPLE
SYSTEM.PASCAL
SYSTEM.MISCINFO
SYSTEM.COMPILER
Im übrigen kann PROTOPAS auf Tricks aller Art zurückgreifen (z.B. „gepokte" Maschinenroutinen ähnlich wie bei PASTOPRO usw.).
Die Zeilen des Quelltextes sollen eine Länge von 79 Zeichen nicht überschreiten, damit man ihn mit einem normalen „SYSTEM.EDITOR" ansehen und ggf. im Peeker ausdrucken kann.
(Die Gesamtlänge von PROTOPAS.TEXT wird ausgewiesen, wenn man ihn mit „Update" oder „Write" speichert.)
2. PROTOPAS.CODE muß sinngemäß ein „normaler" Objektcode sein, der über „Execute" gestartet werden kann. Es versteht sich von selbst, daß wir die eingereichten Programme auf einem „normalen" Apple II testen werden, also keine Accelerator-Karte, kein Z80 mit Turbo-Pascal usw.
(Die Länge von PROTOPAS.CODE kann man über das kleine Programm „LIBRARY.CODE" erfragen, das sich auf der Pascal-Systemdisk Nr. 3 befindet.)

3. Der Einfachheit halber sollte sich die PROTOPAS-Diskette in Slot 6, Drive 1 (Volume „4:") und die ProDOS-Diskette in Slot 6, Drive 2 (Volume „5:") befinden.
4. Nach einem knappen Minimenü (mit Programmname und Name des Programmierers) sollte PROTOPAS dann das ProDOS-*Volume*-Directory selektiv in bezug auf die durchnumerierten TXT-Files anzeigen (siehe oben). Um PROTOPAS nicht unnötig kompliziert zu machen, wird also auf Subdirectories verzichtet.
5. Nach Eingabe der gewünschten Datei-Nummer sollte der Übertragungsvorgang mit einem Piepston und „*** START" beginnen und mit einem Piepston und „*** ENDE" enden, nachdem die konvertierte Ausgangsdatei auf der Zieldiskette gespeichert worden ist. *Vor* „*** START" darf sich bereits das Volume-Directory im Speicher befinden. Dagegen dürfen der Index-Block und die TXT-Datei selbst natürlich erst *nach* „START" eingelesen werden. Das Einlesen und Anzeigen des Directory ist mithin nicht zeitkritisch.
6. Auf der Pascal-Diskette darf noch keine mit „Make" o.ä. im voraus „präparierte" Zieldatei existieren. Damit keine Namenskonflikte auftreten, sollte die Zieldatei den provisorischen Namen „TEMP.TEXT" (für „temporary textfile") erhalten.
7. Die ProDOS-Ausgangsdatei sollte eine Größe von bis zu 32768 Bytes haben dürfen (32K). Mehrfach-Leertasten nach Return sind in die pascal-typischen Tabulatoren umzuwandeln. Beispiel:
Aus 0D 20 20 20 20 20
wird 0D 10 25
„10" steht für Tab und „25" steht für $25 minus $20 = $05 = 5 Leertasten. Die Zieldatei wird damit entsprechend kompakter.
Ctrl-Zeichen (außer $0D) sind zu eliminieren und Bit 7 on ist in Bit 7 off umzuwandeln.
8. Der Zeittest (Zeit zwischen „*** START" und „*** ENDE") wird anhand einer ProDOS-Testdatei namens „TEMP" durchgeführt, die aus 21 Textblöcken besteht und exakt 18360 Bytes umfaßt (siehe Muster Testdatei sowie Miniprogramm, das Testdatei erzeugt). Die Pascal-Zieldatei hat dann nach Tab-Umwandlung einen Umfang von exakt 16840 Bytes und paßt damit gerade noch in den Pascal-Editor-Arbeitsspeicher.
9. PROTOPAS sollte über eine rudimentäre Fehlerabsicherung verfügen. Wenn man beispielsweise anstelle der Datei-Nummer einen String eingibt, so darf das Programm nicht automatisch zusammenbrechen. Eine Absicherung gegen falsch eingelegte Disketten (z.B. DOS 3.3 statt ProDOS usw.), gegen zu große Textfiles (größer als 32K) usw. wäre dagegen zuviel verlangt. Hier ist der Programmabbruch erlaubt.

Nehmen wir als Beispiel ein ähnliches Programm (GETDOS zur Umwandlung von DOS-3.3- in Pascal-Textfiles; siehe Peeker, Heft 1/1985, S. 70ff.). Diese Utility würde den Wettbewerbsbedingungen aus drei Gründen *nicht* genügen:
– Erstens wird kein DOS-Catalog angezeigt. Vielmehr muß man hier den Dateinamen „blind" eingeben (vgl. 4. Bedingung).
– Zweitens werden Mehrfach-Leertasten nicht in „10 XX"-Tab-Sequenzen umgewandelt, womit zu große Zieldateien entstehen (vgl. 7. Bedingung).
– Drittens „verabschiedet" sich das Programm mit „Bad Input" und „Re-initialize", wenn man im Menü mit einem String statt mit einer Zahl antwortet (vgl. 9. Bedingung).

3. Punktesaldo

Die Gewinner des Wettbewerbs werden nach einem „Punktesaldo" ermittelt, der sich aus drei Positionen zusammensetzt:

1. Länge von PROTOPAS.CODE in Bytes
2. Länge von PROTOPAS.TEXT in Bytes
3. Kopierzeit in Sekunden mal 100

Nehmen wir als Beispiel das gelistete Programm PASTOPRO. Der „CODE-File", d.h. das Applesoft-Programm selbst, hat eine Länge von *1330* Bytes. Der „TEXT-File", der normalerweise bei einem Applesoft-Programm gar nicht benötigt wird, hätte eine Länge von *1950* Bytes. Und schließlich dauert der Konvertierungsvorgang für die Testdatei „TEMP.TEXT" 19,5 Sekunden; 19,5 mal 100 = *1950*. Damit ergäbe sich ein „Punktesaldo" von 1330 + 1950 + 1950 = *5230*.

4. Teilnahmebedingungen

Wenn Sie teilnehmen möchten, so senden Sie uns *bis zum 22.07.1985* (Posteingang) eine adressierte Postkarte mit dem Stichwort „Pascal-Wettbewerb" und folgenden vier Zahlen:

xxxx CODE-File-Länge (Bytes)
xxxx TEXT-File-Länge (Bytes)
xxxx Kopierzeit (Sek.) mal 100

---

xxxx Punktesaldo

Die 40 besten Punktesaldo-„Kandidaten" werden dann von uns am 23.07.1985 (Postausgang) per Eilboten gebeten, uns leihweise zur Überprüfung der Werte eine formatierte Pascal-Diskette (35 Spuren) mit dem Volume-Namen „P:" und den Files PROTOPAS.TEXT und PROTOPAS .CODE bis spätestens 01.08.1985 (Posteingang) zuzusenden. Beachten Sie, daß Sie bei der Kopierzeit die Sekundenzahl auf eine ½ Sekunde abrunden können, bevor Sie sie mit 100 malnehmen. Wenn unsere eigenen Überprüfungen einen höheren Punktesaldo ergeben, als auf der Postkarte vermerkt ist, so wird der höhere Punktesaldo eingesetzt. Wenn wir umgekehrt einen niedrigeren Punktesaldo ermitteln, so gilt trotzdem Ihr eigener Postkarten-Punktesaldo. Damit soll ausgeschlossen werden, daß Sie *nach* dem Einsenden der Postkarte noch Programmverbesserungen vornehmen.
Wir werden PROTOPAS.TEXT mit „normalem" Pascal 1.1 und 1.2 zu compilieren versuchen und dann mit Hilfe des neu entstandenen SYSTEM.WRK.CODE die Textdatei übertragen, welche zusätzlich einige Ctrl-Zeichen, Bit-7-on-ASCII-Zeichen sowie etwas „Schrott" im letzten Block enthalten wird.
Die Namen der Gewinner werden dann im Heft 9/1985 mit dem Quellcode des Hauptgewinners veröffentlicht. Sollten zufällig mehrere Gewinner den gleichen Punktesaldo haben, so entscheidet das Los. Im übrigen ist der Rechtsweg ausgeschlossen. Der Hauptgewinner erhält DM 500,-, der Zweitgewinner DM 250,-, und die weiteren Gewinner (3.-25. Preis) erhalten das Buch „ProDOS für Aufsteiger, Band 2" inkl. Begleitdiskette.
U. Stiehl

Pascal-to-ProDOS-Konverter

1. Länge als BIN-File: 1330 Bytes (entspräche Pascal-CODE)
2. Länge als TXT-File: 1950 Bytes (entspräche Pascal-TEXT)
3. Kopierzeit: 19,5s (wird mit 100 multipliziert)
4. Punkte: 5230 (1330 + 1950 + 19,5 * 100)

Die Kopierzeit berechnet sich aus der Differenz zwischen den zwei "Piepstönen" in den Programmzeilen 40-50 von PASTOPRO (*** START bis *** ENDE).

Testdatei

Die Pascal-Testdatei umfaßt 36 Blöcke (brutto) oder 16840 Bytes (netto).
Die ProDOS-Testdatei umfaßt 37 Blöcke (brutto) oder 18360 Bytes (netto). Der Mehrzuwachs ist in den Tabulatoren begründet.

Aufbau der Testdatei:
21 durchnumerierte Textblöcke mit 15 treppenförmig eingerückten Zeilen zu je 50 Zeichen. Unter ProDOS umfaßt 1 Textblock exakt 870 Bytes. 870 * 21 = 18270. Einschließlich der Textblocknummern und der Returns zwischen den Blöcken ergeben sich dann 18360 Bytes.

```
Anfang

1
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
     AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
       AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
         AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
          AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
           AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
             AAAAAAAAAAAAAAAAAAAAAAAAAAAA
              AAAAAAAAAAAAAAAAAAAAAAAAAA
               AAAAAAAAAAAAAAAAAAAAAAAA
                AAAAAAAAAAAAAAAAAAAAAA

2
(Textblöcke 2 bis 21 genauso wie Textblock 1)

Ende
```

Die ProDOS-Testdatei kann mit folgendem Miniprogramm erzeugt werden:

```
100 PRINT CHR$ (4)"OPEN TEMP": PRINT CHR$ (4)"WRITE TEMP"
110 PRINT : PRINT "Anfang": PRINT
120 FOR X = 1 TO 21: PRINT X
130 FOR Y = 1 TO 15
140 S = Y - 1: IF S > 0 THEN FOR Z = 1 TO S: PRINT " ";: NEXT Z
150 FOR Z = 1 TO 50: PRINT "A";: NEXT Z: PRINT
160 NEXT Y: PRINT
170 NEXT X
180 PRINT "Ende": PRINT
190 PRINT CHR$ (4)"CLOSE"
```

PASTOPRO

```
10 DATA 76,28,3,76,63,3,76,129,3,0,0,0,0,3,224,0,0,0,0,32,
   0,191,128,13,3,176,76,96,169,2,141,17,3,169,0,141,18,3,
   169,14,141,16,3,32,19,3,238,16,3,238,16,3,238,17,3,173,
   17,3,201,6
15 DATA 208,237,96,169,14,141,16,3,173,9,3,141,17,3,173,
   10,3,141,18,3,32,19,3,174,16,3,232,232,142,98,3,142,
   16,3,169,255,141,0,16,224,142,144,1,0,238,17,3,208,
   3,238,18,3,173,17,3,205,11,3,144,216
20 DATA 173,18,3,205,12,3,144,208,96,169,0,133,206,169,
   14,133,207,160,0,177,206,240,40,201,255,208,1,96,41,
   127,201,13,240,35,201,32,176,31,201,16,208,21,200,177,
   206,201,33,144,11,170,169,32,32,237,253,202,224,
   32,208,246
25 DATA 32,195,3,32,195,3,76,137,3,32,237,253,76,183,3,
   230,206,208,8,230,207,165,207,201,143,176,152,96:
   FOR X = 768 TO 975: READ Y: POKE X,Y: NEXT:
   IF PEEK (116) < 150 THEN PRINT "HIMEM!": END
30 LOMEM: 36608:M = 768:P = 3584:X = PEEK (48944):
   POKE M + 14,224:D$ = CHR$ (4): PRINT D$"PREFIX,S6":
   ONERR GOTO 70
35 HOME: INVERSE: PRINT "PASTOPRO": PRINT "U.STIEHL": PRINT:
   PRINT "PASCAL";: NORMAL: PRINT "-DISK IN S6,D2": PRINT:
   PRINT "W = WEITER ";: GET X$: ON X$ <> "W" GOTO 35
40 CALL M:O = 0: HOME: GOSUB 55: PRINT: PRINT:
   INPUT "NUMMER: ";X$:O = VAL (X$): ON O = 0 OR O > N
   GOTO 40: HOME: PRINT CHR$ (7)"*** START"
45 GOSUB 55:Y = 256:X = PEEK (B) + PEEK (B + 1) * Y + 2:
   POKE M + 9,X - INT (X / Y) * Y: POKE M + 10, INT (X / Y):
   POKE M + 11, PEEK (B + 2): POKE M + 12, PEEK (B + 3):
   CALL M + 3
50 X$ = "TEMP": PRINT D$"OPEN"X$: PRINT D$"WRITE"X$: PRINT
   CALL M + 6: PRINT: PRINT D$"CLOSE":
   PRINT CHR$ (7)"*** ENDE": END
55 N = 0: FOR X = P TO P + 2047 STEP 26:L = PEEK (X + 6):
   ON L = 0 GOTO 65: ON PEEK (X + 4) < > 3 GOTO 65:
   N = N + 1: IF O = N THEN B = X:X = P + 2047: GOTO 65
60 ON O > 0 GOTO 65: PRINT N" ";: FOR Y = 1 TO L:
   PRINT CHR$ ( PEEK (X + 6 + Y));: NEXT: PRINT
65 NEXT: RETURN
70 PRINT D$"CLOSE": HOME: PRINT "FEHLER " PEEK (222): END
```

PASTOPRO.O

(Gehört nicht zum Wettbewerb, da als DATA-Statements bereits im Applesoft-Programm enthalten.)

```
                        ORG  $0300
                *
                * PASTOPRO.O
                * ==========
                *
                * Pascal-to-ProDOS-Konverter
                * Als DATA-Statements in
                * PASTOPRO enthalten.
                *
                * U.Stiehl/18.05.85
                *
                IND      EQU  $CE
                COUT     EQU  $FDED
                MLI      EQU  $BF00
                *
                * $0E00-$8DFF:Datenpuffer (32K)
                * $8E00      :Endmarker (FF)
                * $8F00      :LOMEM
                *
                * $0E00 = 3584; $8F00 = 36608
                *
                PUFBEG   EQU  $0E00
                PUFEND   EQU  $8E00       ;$8E00-1
                LOMEM    EQU  $8F00
                *
0300: 4C 1C 03  DIR1     JMP  DIR2        ;768=M+0
0303: 4C 3F 03  READ1    JMP  READ2       ;771=M+3
0306: 4C 81 03  WRITE1   JMP  WRITE2      ;774=M+6
0309: 00 00     FIRSTBL  HEX  0000        ;777=M+9
030B: 00 00     LASTBL   HEX  0000        ;779=M+11
                *
                *-----------------------------------
                *
030D: 03        COUNT    HEX  03
                *
                * Unit-Number 60 = S6,D1
                * (Beispiele) E0 = S6,D2
                * (Von Global Page $BF30 = 48944
                * durch Basic-Programm ermittelt.)
                *
030E: E0        UNIT     HEX  E0          ;782=M+14
                *
030F: 00        RDPUFL   HEX  00
0310: 00        RDPUFH   HEX  00
0311: 00        RDBLKL   HEX  00          ;stets 0!
0312: 00        RDBLKH   HEX  00
                *
                * Blockread-Routine
                *
0313: 20 00 BF  RDBLOCK  JSR  MLI
0316: 80                 HEX  80          ;Read
0317: 0D 03              DA   COUNT
0319: B0 4C              BCS  STIRB
031B: 60                 RTS
                *
                *-----------------------------------
                *
                * Directory-Blöcke 2-5 lesen
```

```
                 59  *
031C: A9 02      60  DIR2     LDA  #2          ;$0002
031E: 8D 11 03   61           STA  RDBLKL
0321: A9 00      62           LDA  #0
0323: 8D 12 03   63           STA  RDBLKH
0326: A9 0E      64           LDA  #>PUFBEG
0328: 8D 10 03   65           STA  RDPUFH
032B: 20 13 03   66  DIR3     JSR  RDBLOCK
032E: EE 10 03   67           INC  RDPUFH
0331: EE 10 03   68           INC  RDPUFH
0334: EE 11 03   69           INC  RDBLKL
0337: AD 11 03   70           LDA  RDBLKL
033A: C9 06      71           CMP  #6
033C: D0 ED      72           BNE  DIR3
033E: 60         73           RTS
                 74  *
                 75  *--------------------------------
                 76  *
                 77  * Pascal-Textfile-Blöcke einlesen
                 78  * von FIRSTBL + 2 bis LASTBL - 1.
                 79  * Der FIRSTBL ist bereits vom
                 80  * Applesoft-Programm um 2 erhöht
                 81  * worden.
                 82  *
033F: A9 0E      83  READ2    LDA  #>PUFBEG
0341: 8D 10 03   84           STA  RDPUFH
0344: AD 09 03   85           LDA  FIRSTBL
0347: 8D 11 03   86           STA  RDBLKL
034A: AD 0A 03   87           LDA  FIRSTBL+1
034D: 8D 12 03   88           STA  RDBLKH
0350: 20 13 03   89  READ3    JSR  RDBLOCK
0353: AE 10 03   90           LDX  RDPUFH      ;$8C00
0356: E8         91           INX              ;$8D00
0357: E8         92           INX              ;$8E00
0358: 8E 62 03   93           STX  ENDMARK+2
035B: 8E 10 03   94           STX  RDPUFH
035E: A9 FF      95           LDA  #$FF
0360: 8D 00 10   96  ENDMARK  STA  $1000       ;8E00:FF
                 97  *
                 98  * Wenn < $8E00, dann $8C00
                 99  * wegen 2-Page-Increment.
                100  *
0363: E0 8E     101           CPX  #>PUFEND
0365: 90 01     102           BCC  READ4
                103  *
                104  * "Tödliche" Fehler (BRK)
                105  * a) Blockread-Fehler
                106  * b) Datei > 32K
                107  *
0367: 00        108  STIRB    HEX  00          ;stirb!
                109  *
0368: EE 11 03  110  READ4    INC  RDBLKL
036B: D0 03     111           BNE  READ5
036D: EE 12 03  112           INC  RDBLKH
0370: AD 11 03  113  READ5    LDA  RDBLKL
0373: CD 0B 03  114           CMP  LASTBL
0376: 90 D8     115           BCC  READ3
0378: AD 12 03  116           LDA  RDBLKH
037B: CD 0C 03  117           CMP  LASTBL+1
037E: 90 D0     118           BCC  READ3
0380: 60        119           RTS
                120  *
                121  *--------------------------------
                122  *
                123  * Pascal-Datei über COUT schreiben
                124  *
                125  * Ein Pascal-Textfile hat folgende
                126  * Struktur:
                127  *
                128  * Blöcke 0 + 1:
                129  * -------------
                130  * Systeminformationen, die
                131  * übersprungen werden. Daher ist
                132  * FIRSTBL effektiv FIRSTBL + 2.
                133  *
                134  * Blöcke 2 bis LASTBL - 1:
                135  * ------------------------
                136  * Diese Blöcke enthalten die
                137  * eigentlichen ASCII-Texte. Man
                138  * beachte, daß LASTBL nicht den
                139  * letzten Block DIESES Files,
                140  * sondern den ersten Block des
                141  * NÄCHSTEN Files darstellt.
                142  *
                143  * Doppelblocks
                144  * ------------
```

```
                145  * Der Textfile ab Block 2
                146  * ist in je 2 Doppelblocks
                147  * zu 1024 Bytes zusammengefaßt,
                148  * die am Ende mit Ctrl-0's aufge-
                149  * füllt sein können, auch wenn ein
                150  * weiterer Doppelblock folgt.
                151  * Daher müssen Ctrl-0's stets
                152  * ignoriert werden. Zu dem
                153  * Tab-Problem siehe unten.
                154  *
0381: A9 00     155  WRITE2   LDA  #0
0383: 85 CE     156           STA  IND
0385: A9 0E     157           LDA  #>PUFBEG
0387: 85 CF     158           STA  IND+1
0389: A0 00     159  WRITE3   LDY  #0
038B: B1 CE     160           LDA  (IND),Y
038D: F0 28     161           BEQ  WRITE7      ;Ctrl-0
038F: C9 FF     162           CMP  #$FF        ;Endmark
0391: D0 01     163           BNE  WRITE4
0393: 60        164           RTS              ;Exit
0394: 29 7F     165  WRITE4   AND  #$7F
0396: C9 0D     166           CMP  #$0D        ;Rtn?
0398: F0 23     167           BEQ  WRITE8      ;Ja!
039A: C9 20     168           CMP  #$20        ;Ctrl?
039C: B0 1F     169           BCS  WRITE8      ;Nein!
                170  *
                171  * Tab: $10 $20 nach Rtn entfällt
                172  * Tab: $10 $XX = $XX-$20 Spaces
                173  * (Ctrl-Zeichen außer $10 = Tab
                174  * und $0D = Return ignorieren!)
                175  *
039E: C9 10     176           CMP  #$10        ;Tab
03A0: D0 15     177           BNE  WRITE7
03A2: C8        178           INY              ;IND+1
03A3: B1 CE     179           LDA  (IND),Y     ;Anzahl
03A5: C9 21     180           CMP  #$21
03A7: 90 0B     181           BCC  WRITE6      ;10 20!
                182  *
                183  * Mindestens 1 Space
                184  *
03A9: AA        185           TAX              ;Anzahl
03AA: A9 20     186  WRITE5   LDA  #$20        ;Space
03AC: 20 ED FD  187           JSR  COUT
03AF: CA        188           DEX
03B0: E0 20     189           CPX  #$20
03B2: D0 F6     190           BNE  WRITE5
                191  *
                192  * 10 20 am Zeilenanfang
                193  *^
03B4: 20 C3 03  194  WRITE6   JSR  INC1        ;2mal
03B7: 20 C3 03  195  WRITE7   JSR  INC1        ;1mal
03BA: 4C 89 03  196           JMP  WRITE3
                197  *
                198  * Normale ASCII-Zeichen + Return
                199  *
03BD: 20 ED FD  200  WRITE8   JSR  COUT
03C0: 4C B7 03  201           JMP  WRITE7
                202  *
                203  * Zeiger erhöhen
                204  *
03C3: E6 CE     205  INC1     INC  IND
03C5: D0 08     206           BNE  INC2
03C7: E6 CF     207           INC  IND+1
                208  *
                209  * Wenn IND+1 = LOMEM = $8F00,
                210  * dann Programmabbruch. Dieser
                211  * Fall kann wegen $8E00:FF jedoch
                212  * praktisch nicht vorkommen.
                213  *
03C9: A5 CF     214           LDA  IND+1
03CB: C9 8F     215           CMP  #>LOMEM
03CD: B0 98     216           BCS  STIRB
03CF: 60        217  INC2     RTS
```

208 Bytes

Hinweis

Die Peeker-Sammeldisk enthält die „komfortableren" Versionen PASTOPRO.1D für 1-Drive-Besitzer und PASTOPRO.2D für 2-Drive-Besitzer.

No Orchids für Miss Lisa

Lisa und die Folgen

von Ulrich Stiehl

Als ich im März-Heft mitteilte, daß die Lisa nicht mehr „produziert werden wird" (Futur I), waren offenbar die Würfel bereits gefallen. Ursprünglich gab es von der Lisa mehrere Versionen, nämlich insbesondere Lisa 1 mit Diskettenlaufwerk, Lisa 2 mit 5M-Festplatte und Lisa 2 mit 10M-Festplatte. Zu Beginn dieses Jahres ließ man alle Versionen mit Ausnahme der Lisa 2/10 fallen, die in Macintosh XL umgetauft wurde. Damit sollte zum Ausdruck gebracht werden, daß der Macintosh eigentlich eine kleine Lisa ist. Diese Umbenennung und Verquickung mit dem Macintosh dürfte sich im nachhinein als Bumerang erwiesen haben, weil man sich wenig später dazu entschließen mußte, auch den Macintosh XL = Lisa 2/10 aus dem Programm zu nehmen. Der oberflächliche Beobachter denkt nämlich jetzt, daß mit dem Macintosh XL ein Macintosh eingestellt worden ist. Mutato nomine de te fabula narratur...

Im „Handelsblatt" vom 30. 5. steht, daß wegen der Produktionseinstellung der Lisa über 1600 Arbeitnehmer entlassen werden mußten, was die Börsenspekulanten natürlich nicht unbeeindruckt ließ. Normalerweise wird der „Abgang" eines Mikrocomputers nur noch von den Chronisten der Wirtschaftsblätter vermerkt. Ich halte es jedoch nicht für angebracht, in einer Mikrocomputer-Zeitschrift, die sich ausschließlich mit Apple-Computern befaßt, stillschweigend zur „Tagesordnung" überzugehen. Daß dem „Bürocomputer der Zukunft", wie er in der Werbung in solchen und ähnlichen Slogans apostrophiert wurde, letztlich doch kein Lorbeerkranz geflochten wurde, ist meines Erachtens auf drei Gründe zurückzuführen:

1. Die Lisa war einfach zu teuer. Dies blieb natürlich auch der Firma Apple nicht verborgen, so daß der Anfangspreis von ca. DM 30000,- im Laufe des kurzen und traurigen Lebens der „Mona Lisa" sukzessive gesenkt wurde, bis man zum Frust der wagemutigen Erstkäufer am Ende bei einer Preisreduzierung von ca. 70% angelangt war.
2. Die für den Macintosh XL alias Lisa und den Macintosh konzipierte „Benutzerschnittstelle" mit Ikonen und „Klickiklick" ist nach wie vor heftig umstritten. Hier prallen Welten glühender Verfechter und brüsker Gegner aufeinander. Zu welcher Seite ich gehöre, ist Ihnen ja bekannt.
3. Die Lisa wurde nicht für IBM-(Großanlagen-)kompatibel *gehalten*. Wie so oft im Leben, war hier nicht eine Sache, sondern die *Meinung* über eine Sache entscheidend. Die Lisa wurde – ähnlich wie der Macintosh – von Apple bewußt als IBM-PC-*un*kompatibles Gerät konzipiert. Aber wer glaubte schon, daß man einen IBM-PC-unkompatiblen Mikrocomputer an eine IBM-Mainframe hätte anschließen können. Und wer sollte dies tun? Die Firma IBM etwa? Folgerichtig war dem Macintosh XL der Einzug in die von IBM-Rechnern beherrschte Bürowelt versagt.

In den USA haben praktisch nur noch zwei Mikrocomputer-Produzenten das Sagen, nämlich auf der einen Seite IBM und auf der anderen Seite Apple. Beide vereinigen auf sich einen Marktanteil von weit über 60%, so daß alle übrigen Produzenten bereits „aus dem Rennen" sind. (In Deutschland liegen die Dinge etwas anders, weil hierzulande Commodore nach wie vor eine sehr starke Marktposition hat.) Während jedoch Apple ausschließlich Mikrocomputer produziert, ist für IBM der PC-Bereich bislang noch ein „Nebengeschäft". Seit der Lisa, die vor zwei Jahren auf den Markt kam, versucht Apple mit aller Gewalt, in denjenigen Bereich der Büro-EDV einzudringen, der früher von der mittleren Datentechnik beherrscht wurde. Dabei scheute man sich in den Werbekampagnen des Jahres 1984 nicht, das Orwellsche IBM-Ungeheuer mit der „brave new world" der Lisa und des Macintosh zu kontrastieren. Für Außenstehende stellen sich diese Kontrahenten als ein behäbig schlafender Gulliver und ein hektisch agierender Liliput dar. Immer dann, wenn er von Liliput an der Nase gezupft wird, wälzt sich Gulliver zur Seite und zermalmt dabei irgend etwas unter sich. Heute war es die Lisa. Hoffen wir, daß Liliput morgen etwas mehr aufpaßt.

In dem zitierten Handelsblatt-Beitrag wird bereits das Menetekel der Übernahme durch eine Fremdfirma an die Wand gemalt. Dies erscheint mir abwegig, da Apple nach wie vor ein gesundes Unternehmen ist. Die weitere Entwicklung von Apple hängt im übrigen meines Erachtens weniger vom Erfolg des Macintosh ab. Das Ausscheiden von Steve Wozniak „in gegenseitigem Einvernehmen" zeigt, daß man immer noch nicht einsehen will, daß der Apple II weiterentwickelt werden muß, und zwar so bald wie möglich, denn nach dem Lisa-Debakel ertönte der Gong für die letzte Runde um die Vorherrschaft auf dem amerikanischen Mikrocomputermarkt.

Der Apple II läßt sich mit dem VW Käfer vergleichen, der ebenfalls jahrelang lief und lief. Irgendwann mußte er jedoch durch den VW Golf ersetzt werden, denn die Wünsche der VW-Fahrer waren gewachsen: etwas mehr PS, etwas mehr Komfort, etwas weniger Verbrauch usw. – kurzum alles *etwas* besser. Die Betonung liegt auf „etwas", denn die Firma Apple macht seit geraumer Zeit den Fehler, über das Ziel hinauszuschießen. Wenn wir „im Bild" bleiben, so ist der Macintosh das Elektroauto von übermorgen. Jedermann weiß oder ahnt zumindest, daß in einigen Jahrzehnten Vergaserautos durch Elektroautos verdrängt sein werden. Doch wer kauft heute ein Elektroauto? Und warum nicht? Ähnlich ist es beim Macintosh. Jedermann weiß, daß es in nicht allzuferner

Zukunft nur noch Grafikcomputer mit monströsen Betriebssystemen geben wird. Doch wer kauft heute einen solchen Computer? Und warum nicht?

Was wir *heute* brauchen, ist ein *gradueller* Übergang vom Apple IIe/IIc zum verbesserten „Apple IIg". Einige Beispiele:
1. 80-Zeichenkarte: Die Schriftqualität ist beim Apple IIe brauchbar, aber wie die Sitze des alten VW Käfer verbesserungswürdig. Bit-Map-Grafik wäre keine Lösung für die Gegenwart, denn sonst würde der Apple II wie der Macintosh mit seinen 22K Bildschirmspeicher den Prozessor lahmlegen. Ein sinnvoller Mittelweg wäre eine Schriftqualität in der Art der Ultraterm. Die Firma Videx hat es hier vorgemacht. Apple könnte es nachmachen.
2. Prozessor: Der 1-MHz-65C02 ist brauchbar, aber doch für manche Applikationen zu „käferhaft". Es muß nicht gleich ein 68000 oder ein sonstiger 16- oder gar 32-Bit-Prozessor eingebaut werden. Ein sinnvoller Mittelweg wäre hier ein 65C02C in der Art der Acceleratorkarte. Die Firma Titan hat es hier vorgemacht. Apple könnte es nachmachen.
3. Laufwerke: Ein 140K-Laufwerk ist wie der Kofferraum des alten VW Käfer. Zur Erinnerung: Er war vorne unter der runden Haube. Eine Reihe von Firmen hat gezeigt, daß auch 80-Spur-Laufwerke einwandfrei mit dem Apple II funktionieren können. Apple könnte es nachmachen.

Der Apple II ist deshalb so erfolgreich, weil er „erzkonservativ" ist. Nur wenige möchten ein hypermodernes Gerät kaufen, weil sie mit Recht befürchten, daß es über kurz oder lang wieder vom Markt verschwunden sein wird (siehe Lisa). „Lieber lange einen Macki als kurz einen Mecki", lautet die Devise. Trotzdem muß eine graduelle Weiterentwicklung stattfinden, damit sich der Apple II den sich ändernden Verhältnissen anpaßt. Bleibt zu hoffen, daß die Firma Apple nicht wieder „graduell" mit „abrupt" verwechselt.
(Anmerkung: „No Orchids for Miss Lisa" ist eine Anspielung an das Buch „No Orchids für Miss Blandish" von J.H.Chase. In diesem „Top-10-Krimi" hat sich eine junge Dame namens Blandish (alias Lisa) zum Schluß selbst aus dem Fenster gestürzt, weil sie die Realität nicht mehr ertragen konnte.)

Sie müssen Kalkulationen erstellen ...
Sie müssen Rechnungen schreiben...
Sie müssen schnell ein Angebot zusammenstellen...
Sie brauchen eine zuverlässige Lagerverwaltung
Sie wollen Geschäftsbriefe schreiben
Dann brauchen Sie
HandMac
Das Programm für das Handwerk.
- das Ihnen diese Arbeiten abnimmt
- das speziell für Sie in Deutschland entwickelt wurde
- das Sie sofort benutzen können, ohne mehrwöchige Schulung, ohne seitenstarkes Handbuch

HandMac
Läuft auf Apple Macintosh und gibt es nur im autorisierten Fachhandel
Information: copy team gmbh
schuhstr. 23
8520 erlangen
09131 – 21383
Apple und Macintosh sind eingetragene Warenzeichen der Apple Computer Inc.

Hex-Dez-Konvertierung für 32-Bit-Zahlen

von H. Grumser

Zuweilen stellt sich das Problem, größere Integerzahlen von einem Applesoft-Programm an eine Assemblerroutine zu übergeben. Diese Schwierigkeit tritt z.B. bei der Bearbeitung umfangreicher Random-Access-Dateien auf, bei denen die absolute Dateiposition oftmals 65535 übersteigt. Die USR-Funktion gestattet zwar die Übergabe von Fließkommazahlen. Die Umwandlung in eine Hex-Zahl, die mehr als zwei Bytes umfaßt, ist jedoch mit Hilfe der Interpreter-Routinen nicht mehr möglich.

Die beiden vorgestellten Routinen **FPHEX** und **HEXFP** erlauben die Umwandlung der im Haupt-Fließkomma-Akkumulator abgelegten Zahl in eine 4-Byte-Hex-Zahl und umgekehrt. Diese Zahl kann durch die USR-Funktion an ein Assemblerprogramm übergeben und entsprechend verarbeitet oder ausgewertet werden, um sie dann gegebenenfalls wieder an das Applesoft-Programm zu übertragen.

Beide Routinen sind völlig unabhängig und nicht an ein Applesoft-Programm gebunden. Es wäre somit vorstellbar, die im Applesoft-Interpreter enthaltenen FP-Routinen in einem reinen Binärprogramm zu benutzen; als Schnittstelle zwischen FP- und Integer-Arithmetik könnten dann die beiden Konvertierungsroutinen aufgerufen werden.

Auf die Arbeitsweise der Programme soll hier nur kurz eingegangen werden. Bei der ungepackten Form von FP-Zahlen kann der Integer-Wert im wesentlichen dadurch gefunden werden, daß die Mantisse (FAC $009E–$00A1) so lange verschoben wird, bis der Exponent (FACEXP $009D) den Wert 2 ↑ 32 annimmt. Die vier Mantissenbytes enthalten dann die gewünschte Hex-Zahl. Da die Null nicht in der Mantisse, sondern im Exponenten realisiert wird, muß sie gesondert behandelt werden. Das fünfte Mantissenbyte (FACRND $00AC) dient der genaueren Darstellung bei der Auswertung von Polynomen und darf wegen der hier eventuell aufretenden Linksverschiebung nur eine Null enthalten. Das Vorzeichen (FACSIGN $00A2) bleibt in beiden Fällen unberücksichtigt.

CONVERT

```
                  1   *-----------------------------------------------------*
                  2   *  Convert-Utility für 32-Bit-Integer - Fließkommazahl  *
                  3   *-----------------------------------------------------*
                  4
                  5   VALTYP   EQU  $11        ;Typ des letzten Ausdrucks
                  6   FACEXP   EQU  $9D        ;Exponent von FAC
                  7   FAC      EQU  $9D        ;Haupt-Fließkomma-Akku
                  8   FACSIGN  EQU  $A2        ;FAC-Vorzeichen
                  9   FPTEMP   EQU  $A4        ;Hilfsregister
                  10  FACRND   EQU  $AC        ;FAC-Rundungsstelle
                  11  CHRGET   EQU  $B1        ;Zeichen aus Quelltext holen
                  12  AMPER    EQU  $3F5       ;&-Vektor
                  13
                  14  STROUT   EQU  $DB3A      ;String ausgeben
                  15  OUTDO    EQU  $DB5C      ;Zeichen ausgeben
                  16  MSMERR   EQU  $DD76      ;"TYP MISMATCH ERROR"
                  17  FRMEVL   EQU  $DD7B      ;beliebigen Ausdruck auswerten
                  18  SYNERR   EQU  $DEC9      ;"SYNTAX ERROR"
                  19  NORMFAC  EQU  $E82E      ;FAC normieren
                  20  ZEROFAC  EQU  $E84E      ;0 -> FAC
                  21  OVFLERR  EQU  $E8D5      ;"OVERFLOW ERROR"
                  22  FACSL    EQU  $E8F0      ;FAC-Mantisse Links-Shift
                  23  FOUT     EQU  $ED34      ;FAC in String umwandeln
                  24  PRBYTE   EQU  $FDDA      ;Byte ausgeben
                  25
                  26          ORG  $300
                  27          OBJ  $300
                  28
                  29  * Initialisierung
                  30
0300: A9 03       31          LDA  #>CONVRT   ;&-Vektor
0302: A2 0B       32          LDX  #<CONVRT   ; auf CONVeRT
0304: 8D F7 03    33          STA  AMPER+2    ; setzen
0307: 8E F6 03    34          STX  AMPER+1
030A: 60          35          RTS
                  36
                  37  * Konverierung nach hex oder dezimal
                  38
030B: C9 44       39  CONVRT  CMP  #'D'       ;Dezimalausdruck?
030D: F0 07       40          BEQ  CONDEC
030F: C9 48       41          CMP  #'H'       ;Hex-Zahl?
0311: F0 25       42          BEQ  CONHEX
0313: 4C C9 DE    43          JMP  SYNERR     ;ansonsten "SYNTAX ERROR"
                  44
                  45  * Konvertierung von dezimal nach hex
                  46
0316: 20 B1 00    47  CONDEC  JSR  CHRGET     ;'D' überspringn
0319: 20 7B DD    48          JSR  FRMEVL     ;Ausdruck auswerten
031C: 24 11       49          BIT  VALTYP     ;numerisch?
031E: 10 03       50          BPL  EVLOK      ;ja, dann weiter
0320: 4C 76 DD    51          JMP  MSMERR     ;sonst "TYP MISMATCH"
0323: 20 73 03    52  EVLOK   JSR  FPHEX      ;in 32-Bit-Integer umwandeln
0326: A9 A4       53          LDA  #"$"
0328: 20 5C DB    54          JSR  OUTDO      ;"$" ausgeben
032B: A2 00       55          LDX  #0
032D: B5 9E       56  HEXOUT  LDA  FAC+1,X    ;alle 4 Bytes
032F: 20 DA FD    57          JSR  PRBYTE     ; ausgeben
0332: E8          58          INX
0333: E0 04       59          CPX  #4
0335: D0 F6       60          BNE  HEXOUT
0337: 60          61          RTS
                  62
                  63  * Konvertierung von hex nach dezimal
                  64
0338: A9 00       65  CONHEX  LDA  #0         ;alle
033A: 85 9E       66          STA  FAC+1      ; 4 Bytes
033C: 85 9F       67          STA  FAC+2      ; nullsetzen
033E: 85 A0       68          STA  FAC+3
0340: 85 A1       69          STA  FAC+4
0342: 20 B1 00    70  NXTDIG  JSR  CHRGET     ;Zeichen holen
0345: F0 23       71          BEQ  DECOUT     ;EOS, dann FAC ausgeben
0347: 49 30       72          EOR  #%00110000 ;Bit 4 & 5 invertieren
```

Bei der Benutzung in einem reinen Assemblerprogramm sollte der Sprung nach $E199 durch eine andere Fehlerbehandlungroutine ersetzt werden. Beide Programme sind relokativ.

Das Beispielprogramm **CONVERT** zeigt eine Anwendung der beiden Routinen. Die Umrechnung von Hexadezimal- in Dezimalzahlen ist mühselig und zeitraubend. Mit Hilfe dieser kurzen Ampersand-Utility, die durch „BRUN CONVERT" gestartet wird, kann von Applesoft aus mit „&D" eine Hex-Zahl ausgegeben werden, wobei auf „D" (für dezimal) jeder beliebige Ausdruck, also auch Variablen, folgen kann. Das Ergebnis wird im 8stelligen Hex-Format (32 Bits) ausgegeben. So erhält man z.B. auf die Eingabe „&D 10 * 10" die Ausgabe „$00000064".

Für den umgekehrten Fall ist an Stelle des „D" ein „H", gefolgt von einer bis zu 8 Stellen (= 32 Bits) langen Hex-Zahl, einzugeben. Wird z.B. „&H3E8" eingetippt, erscheint eine Zeile tiefer die Zahl 1000. Die Benutzung hexadezimaler Ausdrücke wie „H3E8 + H64" ist nicht erlaubt.

Beide Befehle können auch im laufenden Programm verwendet werden, wobei darauf hingewiesen sei, daß die Routine selbst keinen Zeilenvorschub (Return) sendet, so daß man mehrspaltige Konvertierungstabellen anlegen kann. Die Ausgabe auf Drucker oder Diskette ist möglich. Die Syntax wurde gewählt, um eine Einbindung in andere Ampersand-Utilities zu ermöglichen.

```
0349: C9 0A      73            CMP  #10        ;Ziffer?
034B: 90 09      74            BCC  SHIFT      ;ja, dann übernehmen
034D: 69 88      75            ADC  #$88       ;in Bereich $FA-$FF bringen
034F: C9 FA      76            CMP  #$F0+10    ;war Zeichen zw. 'A'-'F'?
0351: B0 03      77            BCS  SHIFT      ;ja, dann übernehmen
0353: 4C C9 DE   78            JMP  SYNERR     ;ansonsten "SYNTAX ERROR"
0356: 0A         79   SHIFT    ASL             ;rechtes Nibble
0357: 0A         80            ASL             ; nach links
0358: 0A         81            ASL             ; schieben
0359: 0A         82            ASL
035A: A2 03      83            LDX  #3         ;um ein Nibble
035C: 0A         84   NXTBIT   ASL             ; nach links
035D: 26 A1      85            ROL  FAC+4      ; (sowohl Akku
035F: 26 A0      86            ROL  FAC+3      ;  als auch
0361: 26 9F      87            ROL  FAC+2      ;  FAC)
0363: 26 9E      88            ROL  FAC+1
0365: CA         89            DEX
0366: 10 F4      90            BPL  NXTBIT     ;nächstes Bit
0368: 30 D8      91            BMI  NXTDIG     ;nächstes Zeichen
036A: 20 94 03   92   DECOUT   JSR  HEXFP      ;in FP-Zahl umwandeln
036D: 20 34 ED   93            JSR  FOUT       ;in String umwandeln
0370: 4C 3A DB   94            JMP  STROUT     ;String ausgeben
                 95
                 96   * FAC -> 32-Bit-Integer ($9E - $A1)
                 97
0373: A5 9D      98   FPHEX    LDA  FACEXP     ;FAC < 1?
0375: 30 0B      99            BMI  NOTZER     ;nein, dann weiter
0377: A9 00      100           LDA  #0         ;ansonsten
0379: 85 9E      101           STA  FAC+1      ; alle 4 Bytes
037B: 85 9F      102           STA  FAC+2      ; nullsetzen
037D: 85 A0      103           STA  FAC+3
037F: 85 A1      104           STA  FAC+4
0381: 60         105           RTS             ;fertig
0382: C9 A1      106  NOTZER   CMP  #$80+33    ;größer 2 ↑ 32?
0384: 90 03      107           BCC  NOOVFL     ;nein, dann weiter
0386: 4C D5 E8   108           JMP  OVFLERR    ;sont "OVERFLOW"
0389: 69 60      109  NOOVFL   ADC  #$80-32    ;negativer Wert von Shifts
038B: A0 00      110           LDY  #0
038D: 84 A4      111           STY  FPTEMP     ;FP-Hilfsregister
038F: A2 9D      112           LDX  #FAC       ;Stellenwertigkeit
0391: 4C F0 E8   113           JMP  FACSL      ; berichtigen
                 114
                 115  * 32-Bit-Integer ($9E - $A1) -> FAC
                 116
0394: A9 A0      117  HEXFP    LDA  #$80+32    ;Exponent
0396: 85 9D      118           STA  FACEXP     ; setzen
0398: A9 00      119           LDA  #0         ;Rundungstelle
039A: 85 AC      120           STA  FACRND     ; nullsetzen
039C: 85 A2      121           STA  FACSIGN    ;pos. Vorzeichen
039E: A5 9E      122           LDA  FAC+1      ;alle
03A0: 05 9F      123           ORA  FAC+2      ; Stellen
03A2: 05 A0      124           ORA  FAC+3      ; Null?
03A4: 05 A1      125           ORA  FAC+4
03A6: F0 03      126           BEQ  ISZER      ;ja, dann Sonderbehandlung
03A8: 4C 2E E8   127           JMP  NORMFAC    ;normieren, fertig
03AB: 4C 4E E8   128  ISZER    JMP  ZEROFAC    ;FAC nullsetzen

174 Bytes
```


Peeker-Sammeldisk #7

zu Heft 7/1985, einzeln DM 28,–
im Fortsetzungsbezug DM 20,–

A 004 PYRAMID.PITTY
T 084 T.PYR.PITTY.0
T 063 T.PYR.PITTY.1
B 018 PYR.PITTY.0
B 016 PYR.PITTY.1
B 024 PYR.PITTY.BACK
B 026 PYR.PITTY.SHAPE

T 016 T.MEGAWARP.REL
B 002 MEGAWARP.REL

T 017 T.MEGAWARP.9900
B 002 MEGAWARP.9900
T 004 T.SPEEDTEST
B 002 SPEEDTEST

A 016 FORMAT
T 039 T.FORMAT.OBJ
B 005 FORMAT.OBJ

A 033 BITEDITOR
B 018 NORMAL
B 018 FETT
B 018 FETT.INVERSE

A 007 PASTOPRO.1D
A 007 PASTOPRO.2D
T 015 T.PASTOPRO.O
B 002 PASTOPRO.O

T 014 T.CONVERT
B 002 CONVERT

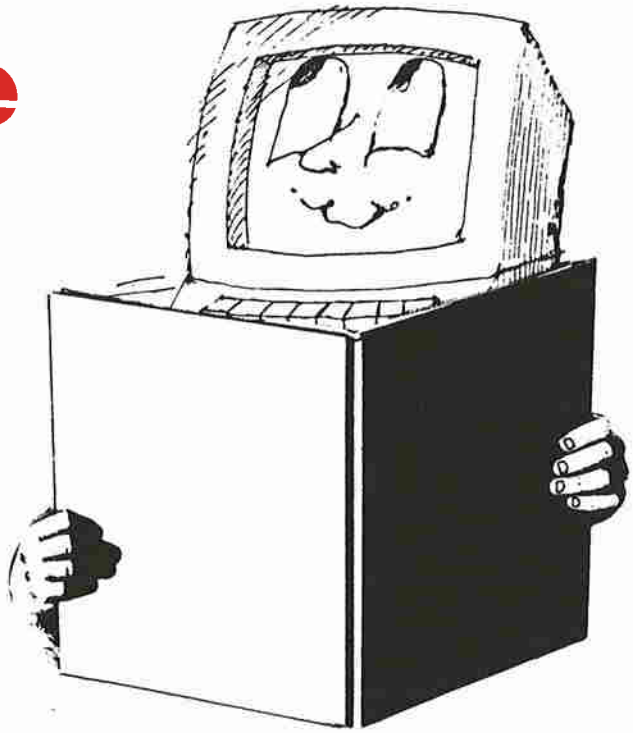
T 010 T.VORLESER
B 003 VORLESER

Hüthig Software Service · Postfach 10 28 69 · 6900 Heidelberg

Vorlesestunde

Apple und S.A.M., ein hilfreiches Gespann

von Dr. Jürgen B. Kehrel



Sie haben aus einer Zeitschrift ein Assemblerlisting oder einen Hex-Dump mühsam abgetippt, doch irgendwo hat sich ein Fehler eingeschlichen. Jetzt müssen Sie Ihren Speicherauszug mit dem geschriebenen Text vergleichen. Ihr Blick wandert zwischen Bildschirm und Papier hin und her, der Zeigefinger hält die Position fest. Doch immer wieder kommen Sie aus dem Tritt. – Wenn Sie diese Situation jemals kennengelernt haben, ist das folgende Programm die Erlösung von den erlebten Qualen.

Im Peeker 2/84 wurde die Sprachkarte S.A.M. (Software Automatic Mouth) von „Don't Ask Software" vorgestellt, durch die Sie beliebige (vornehmlich englische) Texte mit Hilfe der zugehörigen Software über einen Lautsprecher ausgeben können. Ich kam bald auf den Gedanken, mir die Speicherauszüge von S.A.M. vorlesen zu lassen. Ein kurzes Assemblerprogramm und die RECITER-Software machten dies schnell möglich, doch waren die Buchstaben A–F sehr schlecht verständlich. Außerdem belegten S.A.M. und RECITER fast den halben Speicher. Mein nächster Versuch wurde länger, benötigte dafür aber nur noch das S.A.M.-Modul, da ich diesmal Phoneme benutzte.

S.A.M. liegt ab $7161 im Speicher, also assemblierte ich meinen Teil direkt darunter.

Wenn Sie nacheinander „VORLESER" und „SAM" laden (BLOAD), können Sie beide zusammen mit „BSAVE DUMPVORLESER, A$7050, L$24B0" abspeichern und später als einen funktionsfähigen File mit „BRUN DUMPVORLESER" starten.

Ihre Speicherauszüge erhalten Sie weiterhin so, wie es im Benutzerhandbuch des Apple beschrieben wird: (hexadezimale) Startadresse.Endadresse <Return>. Zusätzlich zur Bildschirmausgabe spricht S.A.M. nun alle Hex-Zahlen mit.

Programmbeschreibung

Alle Ausgaben des Apple werden normalerweise durch das DOS gelenkt, bevor sie z.B. zum Bildschirm gelangen. Genau an dieser Stelle schaltet sich „VORLESER" dazwischen. Das Startprogramm überträgt den Beginn des Hauptprogramms (Adresse von VEKTOR) in den DOS-Ausgabe-Vektor (nur DOS 3.3, nicht ProDOS!) und springt in den Monitor. Bei allen „Nicht-DOS"-Ausgaben verzweigt nun das Hauptprogramm selber zur Bildschirmausgabe (JSR COUT1), versorgt aber zusätzlich S.A.M. mit den nötigen Informationen. Punkte, Striche, Leerzeichen und Returns werden in Pausen umgewandelt, um die Verständlichkeit zu erhöhen. Sie können die Pausen vergrößern oder verkleinern, indem Sie nach $70D7 in den LDA-Befehl einen größeren oder kleineren Wert als $20 schreiben.

Alle gültigen Hex-Zahlen werden mit einer Routine herausgefiltert, die ich teilweise dem Apple-Monitor ($FFB1–$FFBC) entliehen habe. Alle übrigen Zeichen werden nicht bearbeitet (EXIT). Durch die Überprüfung werden gleichzeitig die Hex-Zahlen „0" bis „F" in die Bytes $00 bis $0F umgeformt. Da jeder Tabelleneintrag der Phoneme 8 Bytes lang ist, werden die Zahlenwerte für die indizierte Adressierung mit 8 multipliziert. Vorher wird noch eine 1 addiert, da die Einträge rückwärts gelesen werden. Der passende Tabellenteil wird in den S.A.M.-Puffer nach $9500 übertragen und dann durch JSR SAM ausgesprochen. Wenn Ihnen irgendeine Aussprache nicht gefällt, können Sie den Tabelleneintrag ändern, solange die Summe der voranstehenden Nullen und der Phoneme gleich 8 ist. Maximal sind 8 Phoneme ohne Null zulässig.

Die Befehle am Anfang und Ende des Programms retten nur die relevanten Register und Speicherstellen, da S.A.M. sie durch seinen Betrieb zerstört.

So, und nun wünsche ich Ihnen erfolgreiche Vorlesestunden.

VORLESER

```
                  1   ******************************
                  2   *          VORLESER          *
                  3   *  von Dr. Jürgen B. Kehrel  *
                  4   *            1985            *
                  5   ******************************
                  6   *
                  7   * Liest Hex-Dumps des Applespei-
                  8   * chers vor, wenn in Slot 4 eine
                  9   * S.A.M-Karte vorhanden und das
                  10  * Modul SAM ab $7161 geladen ist.
                  11  *
                  12  * Vorleser liegt direkt unter SAM
                  13  *
                  14          ORG  $7050
                  15  *
                  16  PUFFER   EQU  $9500      ;Ausgabep.
                  17  COUT1    EQU  $FDF0      ;Bildschirm
                  18  WAIT     EQU  $FCA8      ;Zeitverzög.
                  19  MONZ     EQU  $FF69      ;Monitor
                  20  SAM      EQU  $94F6      ;Einsprung
                  21  A1       EQU  $3C        ;Zero-Page
                  22  *
7050: 4C 5A 70    23          JMP  START
                  24  *
7053: 00          25  ASAVE    HEX  00         ;Register
7054: 00          26  XSAVE    HEX  00         ;und div.
7055: 00          27  YSAVE    HEX  00         ;Speicher
                  28  A1SAVE   DS   $4         ;retten
                  29  *
705A: D8          30  START    CLD             ;Binärmode
705B: A9 68       31          LDA  #<VEKTOR
705D: 8D 53 AA    32          STA  $AA53      ;DOSCSWL
7060: A9 70       33          LDA  #>VEKTOR
7062: 8D 54 AA    34          STA  $AA54      ;DOSCSWH
7065: 4C 69 FF    35          JMP  MONZ
                  36  *
                  37  *
7068: 8D 53 70    38  VEKTOR   STA  ASAVE      ;retten
706B: 8E 54 70    39          STX  XSAVE
706E: 8C 55 70    40          STY  YSAVE
7071: A2 03       41          LDX  #3
7073: B5 3C       42  LOOP     LDA  A1,X
7075: 9D 56 70    43          STA  A1SAVE,X
7078: CA          44          DEX
7079: 10 F8       45          BPL  LOOP
707B: AD 53 70    46          LDA  ASAVE
707E: 20 F0 FD    47          JSR  COUT1      ;ausgeben
7081: 09 80       48          ORA  #$80       ;Bit 7
7083: C9 8D       49          CMP  #$8D       ;Return
7085: F0 4D       50          BEQ  WARTEN
7087: C9 A0       51          CMP  #$A0       ;" "
7089: F0 49       52          BEQ  WARTEN
708B: C9 AD       53          CMP  #$AD       ;"-"
708D: F0 45       54          BEQ  WARTEN
708F: C9 AE       55          CMP  #$AE       ;"."
7091: F0 41       56          BEQ  WARTEN
7093: 49 B0       57          EOR  #$B0
7095: C9 0A       58          CMP  #$0A       ;0 - 9
7097: 90 09       59          BCC  ZAHL
7099: 69 88       60          ADC  #$88       ;C = 1 !
709B: C9 FA       61          CMP  #$FA       ;A - F
709D: 90 21       62          BCC  EXIT
709F: 29 0F       63          AND  #$0F       ;%00001111
70A1: 18          64          CLC
70A2: 69 01       65  ZAHL     ADC  #$01       ;Tabellenpos.
70A4: 0A          66          ASL             ;ausrechnen
70A5: 0A          67          ASL
70A6: 0A          68          ASL             ; x 8
70A7: A8          69          TAY
70A8: A2 00       70          LDX  #$00
70AA: 88          71  LADEN    DEY
70AB: B9 E0 70    72          LDA  TAB,Y
70AE: F0 08       73          BEQ  SPRECHEN   ;bis $00
70B0: 9D 00 95    74          STA  PUFFER,X
70B3: E8          75          INX
70B4: E0 08       76          CPX  #$08
70B6: 90 F2       77          BCC  LADEN
70B8: A9 8D       78  SPRECHEN LDA #$8D       ;Return
70BA: 9D 00 95    79          STA  PUFFER,X
70BD: 20 F6 94    80          JSR  SAM
70C0: A2 03       81  EXIT     LDX  #$03       ;Speicher u.
70C2: BD 56 70    82  EX1      LDA  A1SAVE,X   ;Register zu-
70C5: 95 3C       83          STA  A1,X       ;rückschreiben
70C7: CA          84          DEX
70C8: 10 F8       85          BPL  EX1
70CA: AD 53 70    86          LDA  ASAVE
70CD: AE 54 70    87          LDX  XSAVE
70D0: AC 55 70    88          LDY  YSAVE
70D3: 60          89          RTS             ;Ende
                  90  *
70D4: A2 C8       91  WARTEN   LDX  #$C8       ;Warte-
70D6: A9 20       92  WART1    LDA  #$20       ;schleife
70D8: 20 A8 FC    93          JSR  WAIT
70DB: CA          94          DEX
70DC: D0 F8       95          BNE  WART1
70DE: F0 E0       96          BEQ  EXIT
                  97  *
                  98  * Liste der Phoneme
                  99  * jeweils 8 Bytes lang
                  100 *
70E0: 00          101 TAB      HEX  00
70E1: D7 CF D2    102         ASC  "WOR5YIZ"  ;Zero
70E4: B5 D9 C9 DA
70E8: 00 00 00    103         HEX  000000
70EB: CE B5 C8    104         ASC  "N5HAW"    ;One
70EE: C1 D7
70F0: 00 00 00    105         HEX  00000000
70F3: 00
70F4: B5 D7 D5    106         ASC  "5WUT"     ;Two
70F7: D4
70F8: 00 00       107         HEX  0000
70FA: B5 D9 C9    108         ASC  "5YIRHT"   ;Three
70FD: D2 C8 D4
7100: 00 00 00    109         HEX  000000
7103: D2 B5 C8    110         ASC  "R5HOF"    ;Four
7106: CF C6
7108: 00 00 00    111         HEX  000000
710B: D6 B5 D9    112         ASC  "V5YAF"    ;Five
710E: C1 C6
7110: 00 00       113         HEX  0000
7112: D3 CB B5    114         ASC  "SK5HIS"   ;Six
7115: C8 C9 D3
7118: CE D8 C9    115         ASC  "NXIV5HES" ;Seven
711B: D6 B5 C8 C5 D3
7120: 00 00       116         HEX  0000
7122: D4 D4 D4    117         ASC  "TTT6YE"   ;Eight
7125: B6 D9 C5
7128: 00 00 00    118         HEX  000000
712B: CE B5 D9    119         ASC  "N5YAN"    ;Nine
712E: C1 CE
7130: 00 00 00    120         HEX  000000
7133: B6 C8 C9    121         ASC  "6HIYE"    ;A
7136: D9 C5
7138: 00 00       122         HEX  0000
713A: B5 D9 C9    123         ASC  "5YIYIB"   ;B
713D: D9 C9 C2
7140: 00 00       124         HEX  0000
7142: B5 D9 C9    125         ASC  "5YIYIS"   ;C
7145: D9 C9 D3
7148: 00          126         HEX  00
7149: B5 D9 C9    127         ASC  "5YIYIDD"  ;D
714C: D9 C9 C4 C4
7150: 00 00 00    128         HEX  000000
7153: B5 D9 C9    129         ASC  "5YIYI"    ;E
7156: D9 C9
7158: 00 00 00    130         HEX  00000000
715B: 00
715C: C6 B5 C8    131         ASC  "F5HE"     ;F
715F: C5

272 Bytes
```

Nachtrag zu RAM.FRE

Das Programm RAM.FRE aus Peeker 1/2-85, Seite 33 enthält einen schwerwiegenden Fehler und läuft daher nur in wenigen Fällen korrekt.
Die fehlerfreie Version wird im nächsten Peeker erscheinen.

Leserbriefe

DOS-Mover

Gratuliere zur Peeker! Ich habe noch keine bessere Apple-Zeitschrift gesehen. Machen Sie weiter so! Ein Thema was mich besonders interessiert, ist das Verschieben des DOS in die LC: ein DOS-Mover, in allen Einzelheiten erklärt.
Ivo Eschrich, Göttingen
(Ein neuer DOS-Mover für Standard-DOS 3.3 ist in Vorbereitung. us)

BASIS 108

Nachdem ich mit meinem ITT-2020 nicht gerade völlig zufrieden war, besorgte ich mir 1982 einen BASIS 108 (Hersteller Basis Microcomputer in Münster; der ehemalige Apple-Generalimporteur). Für jene, die das Gerät noch nicht kennen: Dieser Kompatible hat 130K RAM, 12K ROM (nicht mitgeliefert), parallele und serielle Schnittstellen, Z80, 80 Zeichen, SW-, PAL- und RGB-Ausgang, Platz für 2 „dicke" Laufwerke im Gehäuse, Tastatur mit Cursorblock, numerischem Block, 15 4fach belegbaren Funktionstasten, sowie die Möglichkeit, IRQs zu erzeugen. Trotzdem sind noch die Slots 2-7 frei verfügbar. Leider ist deutsche Qualität nicht ganz billig. Ich rüstete den Computer mit den damals noch fehlenden zweiten 64K RAM aus, setzte mir die Applesoft-EPROMs ein und war völlig zufrieden. Als sich ein Bekannter später ebenfalls solch ein Gerät zulegte, fiel mir dessen Betriebsanleitung auf (ähnlich „Apple Benutzer Handbuch"). Da ich nur eine vorläufige Ausgabe bekommen hatte, fragte ich bei Basis an, ob ich solch ein Exemplar kaufen könnte. Wenige Tage später bekam ich kostenlos eines geliefert. Leider häuften sich später die Gerüchte, daß die Firma nicht mehr existieren würde. Das mag kein Computerbesitzer gerne hören. Von einem Basis-Besitzer aus Münster erfuhr ich vor knapp einem Jahr, daß es die Firma unter neuem Namen einige Straßen weiter wieder gibt. Sie heißt jetzt Basis Computersysteme GmbH. Es störte mich schon einige Zeit, daß bei dem Drucker-PROM die Tabulierung nicht richtig funktionierte. Die jüngeren Exemplare hatten diesen Bug nicht mehr. Leider konnte ich es nicht selbst ersetzen, da ich niemanden kenne, der schmale PROMs mit 20 Pins (ähnlich Controller-PROMs) brennen kann. Außerdem war mein F8-EPROM noch nicht in der Lage, softwaremäßig zwischen 40 und 80 Zeichen umzuschalten. Da ich den Computer nur mit 66K RAM erhalten hatte, fehlte mir noch die Pseudodisk-Software für das UCSD-System. Diese Probleme beschrieb ich in einem Brief an die „neue" Firma Basis mit der Frage, ob ich die Bauteile und die Software erwerben könne. Eine halbe Woche später fand ich im Briefkasten einen Umschlag, in welchem neben einem Begleitschreiben bereits eine ZAP-Diskette mit der gewünschten Software und die beiden PROMs waren! Ich finde, daß eine derartige Kulanz Vorbild sein sollte für viele andere Firmen (inkl. Apple selbst).
Tim Berndt, Büdelsdorf

Erphi-Controller

Zu Ihrer Frage in Peeker 2/84, ob bei Testberichten auch über die Schattenseiten des Produkts berichtet werden sollte, meine ich, daß dies unbedingt geschehen sollte. Nur wenn die positiven und negativen Seiten des jeweiligen Produkts beschrieben werden, ist der potentielle Käufer vor unliebsamen Überraschungen einigermaßen sicher. Leider werden die Schattenseiten des Erphi-Controllers in Peeker 4/85 nicht erwähnt. Deshalb möchte ich hier über die Nachteile dieses Controllers berichten. Die höhere Speicherkapazität der Laufwerke kann nur ausgenutzt werden, wenn der Controller in Slot 6 steckt und auch von Slot 6 gebootet wird. Ist dies nicht der Fall, können nur 35 Tracks verwendet werden. Da ich von einer an Slot 7 angeschlossenen Harddisk bootete, konnte ich auf Floppy nur lächerliche 140K verwenden. Bei den Floppy-Controllern, bei denen das Image der Betriebssysteme auf der Diskette angepaßt wird, treten diese Probleme nicht auf, da man das angepaßte Betriebssystem auf die Harddisk kopieren kann, was bei dem Erphi-Controller nicht geht. Auch die Hardware dieses Controllers ist problematisch. Es werden an der Schnittstelle zu den Standardlaufwerken (Shugart-Bus) Leitungstreiber verwendet, die den bei 150 Ohm Pull-Up-Widerständen notwendigen Strom nicht liefern können. Es wird empfohlen, Pull-Up-Widerstände mit 220 Ohm oder mehr zu verwenden. Es wird jedoch verschwiegen, wie ein im Laufwerk eingelötetes Netzwerk mit 150 Ohm gegen 220 Ohm ausgetauscht werden kann.
Ulrich Allgeier, Stuttgart

IIe-80-Z/Z-Karte

Ich bin seit März 1983 ein mehr oder weniger zufriedener Besitzer eines Apple IIe. In technischen Fragen wurde ich von meinem Händler immer sehr gut beraten. Lediglich auf die Frage nach der doppelt hochauflösenden Grafik, die ich nach Lektüre eines Artikels in der Zeitschrift „nibble" von 8/84 stellte, wußte er nicht weiter und gab mir die Telefonnummer der Firma Apple.
Nach einer peinlichen Überprüfung, ob ich denn einen „Original-Apple" hätte, wo ich ihn gekauft hätte (hoffentlich ein autorisierter Händler!) usw. versprach der PR-Mann mir, daß er mich nach einer Beratung mit einem Techniker zurückrufen werde. Beim Rückruf wurde mir erklärt, Apple-IIe-Rechner Revision A deutsch wären nicht ausgeliefert worden. Auf meinen Einwand und der Durchsage der Gerätenummer wurde ein erneuter Rückruf ausgemacht. Bei diesem Rückruf wurde mir dann erklärt, welche Pins zu verbinden wären und daß die deutsche Revision A (die es dann also doch gab) identisch mit der amerikanischen Revision B sei. Anschließend wurde mir nahegelegt, in Zukunft doch deutsche Zeitschriften zu lesen.
Für die Zukunft würde ich mich über weitere Artikel freuen, die die Nutzung der erweiterten 80-Zeichenkarte in eigenen Programmen (Zwischenspeicherung größerer Matrizen etc.) beschreiben.
Gerhard Hübschle, Stuttgart
Ein solches Programm speziell für Zahlen-Arrays wird in einem der nächsten Peeker-Hefte abgedruckt. us.

16K AKKU-RAM-KARTE

* Kompatibel zur Standard 16K Language Card
* Datensicherung nach Abschalten
* PSEUDO-ROM (z.B Rom-Edit etc.)
* leichter Einbau
* fuer alle Slots verwendbar

175.-DM

128K AKKU-RAM-KARTE mit Software 795.-DM

16K STANDARD	100.-
80Z KARTE mit Softswitch	170.-
Z80 KARTE	100.-
128K KARTE mit Software	350.-
256K KARTE mit Software	495.-
512K KARTE mit Software	795.-

Made in Germany

Alle Preise incl 14% MWST
zuzuegl. Porto + Verpackung
Info Gratis
Haendleranfragen erwuenscht

Ing.Buero M.Fricke
Neue Str.13
1000 Berlin 37
Tel. 030/8015652

Ausgabe und Eingabe mit TYPETERM®
im Slot Ihres
APPLE II/IIe

Das bedeutet: Computertextverarbeitung von der Schreibmaschinentastatur! Steckerfertig ohne Umbau.

TYPETERM-Interface DM 479,– incl. MWSt.
für alle BROTHER-Typenradschreibmaschinen

Paketpreis: DM 1348,–
Schreibmaschine CE-51 mit TYPETERM

brother
QUALITÄT AUS ERSTER HAND.

CE-61 mit TYPETERM ..... DM 1787,–
EM-80 mit TYPETERM ..... DM 2087,–
EM-100 mit TYPETERM ..... DM 3122,–
TYPETERM-Kit für CE-50 ..... DM 468,–
CE-25 mit TYPETERM ..... anfragen

TYPETERM – ein starkes Interface für starke Maschinen! Alle Cursor- und Ctl-Befehle. 2k ROM auf der Karte f. DOS, PRODOS, CP/M, PASCAL. Alle Features: Hoch-/Tiefstellen, autom. Unterstreichen, var. Zeichen- u. Zeilenabst., autom. Papierzuführung usw. Ausführl. Handbuch vorab: 10,– DM auf Konto 14770-306 PGiroA Han (Anrechnung).

TYPETERM ein Produkt von
interkom electronic
Kock & Mreches GmbH
Postf., 3004 Isernhagen 4
Telefon 05139-87393

Schule und Peeker
Zu Ihrer Zeitschrift möchte ich Ihnen gratulieren. Sie hebt sich angenehm von den meisten übrigen Blättern auf diesem Sektor ab. Was mir gefällt möchte ich kurz auflisten:
– Ihre Artikel vermitteln den Eindruck der Kompetenz des Autors.
– Sie bereiten nichts so weit „redaktionell" auf, bis hinter bunten Bildern und flotter Sprache nur noch Plattheiten versteckt sind, die dem Anfänger nichts nützen und den Fortgeschrittenen ärgern.
– Ihre Autoren haben bei Testberichten den Mut zu eigener Meinung und lassen ihr Urteil nicht in wolkenreicher Watteverpackung verschwinden.
– Sie vermitteln nicht den Eindruck einer Werbeschrift für die Verbreitung von Microcomputern, gleich welcher Firma.
Was mir noch fehlt und was ich noch gerne finden würde kommt aus meiner besonderen Interessenlage. Ich unterrichte an der Sekundarstufe I Mathematik und Physik und im Wahlfach Informatik. Dabei habe ich mir ein Grundlagenwissen erworben. So liegt z.B. der Artikel über die Accelerator IIe noch außerhalb meines Verständnishorizontes. Soweit ich in meinem Umfeld beobachten kann, breitet sich die Verwendung des Apples im Schulbereich aus, und die meisten Lehrer sind keine Spezialisten, aber doch potentielle Leser und Abonnenten. Ihre Rubrik Schule könnten Sie unter diesem Gesichtspunkt ausbauen. Dabei sollte der Bereich vor dem Programmlisting besser aufbereitet werden. Ich denke an Struktogramme und Programmablaufpläne. Es wird dadurch für den Nicht-Profi-Lehrer leichter auswertbar, z.B. im Peeker 3/85 die „Multiprecision". Mir wäre *eine* Rechenart lieber, die aber in dem *gleichen* Umfang ausführlich dargestellt wird. Unter dieser Rubrik Schule sollte sich ein Autor nicht für „eher primitive" Techniken quasi entschuldigen. Die Kunst des Lehrens ist es, etwas einfach und einsichtig zu machen, was man selbst kann und einem einfach vorkommt, verglichen mit dem Wissen und Können von Spezialisten. Für Lernende ist nach meiner eigenen und auch der Erfahrung mit Schülern die Aufbereitung einer Aufgabenstellung, bis sie „programmierbar" wird, der schwierigste Teil des Weges. Sie sind hier sicher auch auf die Hilfe von Autoren angewiesen. Vielleicht könnten Sie hierfür auch vorhandene Literatur auswerten, was für eine Fachzeitschrift auch verdienstvoll ist, da Nicht-Profis nicht den Überblick haben können. Insgesamt unterstütze ich Ihre Konzeption und habe deswegen Ihre Zeitschrift abonniert.
K.H.Ruppert, Würzburg

Weiter so!
Nach intensivem Studium von Peeker 3/85 und 4/85 habe ich mich zum Abonnement entschlossen. Mit dieser Zeitschrift haben Sie wirklich „den Nagel auf den Kopf getroffen", d.h. endlich steht einmal sachliche Information vor dem seichten Nachpredigen von Werbeschriften der Computerhersteller. Sehr gut: Endlich macht jemand Apple deutlich, warum der Apple II so erfolgreich ist und die neueren Produkte bei dieser Verkaufspolitik Krücken sind und bleiben. Als Anwender muß ich klar sagen, daß es von Apple selbst zum IIc keine Alternative gibt.
Die Devise lautet: Machen Sie weiter so!
Dipl.Ing.G.Fischer, Holzmaden

Nicht weiter so!
Auch als Noch-Nicht-Abonnent Ihres Peeker sehe ich mit einigem Interesse der nächsten Ausgabe entgegen, wird er doch sicherlich einiges für mich Nützliche enthalten – vielleicht sogar im redaktionellen Teil.
Wie Sie sehen, ist etwas mit Ironie und Überheblichkeit sehr schnell gesagt, oder in unserem Fall geschrieben, doch Stil ist das – Sie werden mir beipflichten – ganz bestimmt nicht. Doch warum plagt mich eigentlich seit der ersten Ausgabe das Gefühl, daß hier ein Chefredakteur in für mich persönlich zunehmend unerträglicher Weise sein Podium benutzt, um ständig zu betonen, wie herausragend gut man selber und wie lächerlich das Bemühen anderer ist? Nun gibt es den Apple ja schon

APPLE II kompatibles

PC-48 (europlus) **799,-**

PC-64 (europlus + 16K) **899,-**

DISTAR-Drive f. alle II-Typen
II, IIe **439,-**
IIc **457,-**

Info gegen DM 1,40 in Briefmarken
SPRINGMANN Computer GmbH
Stöckener Str. 199
3000 Hannover 21
05 11/79 11 11

eine ganze Weile, und der Kreis derer, die diesen Rechner in- und auswendig kennen, ist erfreulicherweise so groß, daß man die Zugehörigkeit nicht ebenfalls ständig elitär vor sich hertragen muß.
Resultiert Ihr unzweifelhaft existenter Wissensvorsprung in Apple-Angelegenheiten vielleicht nur aus der Tatsache, daß sich Ihre Leser nicht hauptberuflich mit diesem Thema beschäftigen können und einmal monatlich die Brosamen aufpeeken, die da vom Redaktionstisch fallen?
Trotzdem, die Anzahl der fundierten Aufsätze und das Themenspektrum lassen mich über gewisse (Un)Feinheiten hinwegsehen. Daß ich der rein fachlichen Information Vorrang einräume, wollen Sie bitte beiliegender Abonnementbestellung entnehmen. Mit meiner Meinung möchte ich allerdings nicht hinter dem Berg bleiben, obwohl ich fast nicht annehme, daß Sie Ihrer eher Pro ausgerichteten Leserbriefseite ein wohltuendes Gegengewicht verleihen wollen – oder doch, man weiß ja nie.

Helge Baars

Es ist richtig, daß ich mich gelegentlich über Mißstände mockiere, und dies werde ich auch weiterhin tun, denn nur wenn man einen Fehler ins Rampenlicht zerrt, besteht die Hoffnung, daß er auch beseitigt wird. Dabei liegt es mir allerdings fern, meine Leser durch den „Kakao zu ziehen". Sollte dieser Eindruck bei Ihnen entstanden sein, so möchte ich mich nachdrücklich bei Ihnen entschuldigen. Nehmen wir, da Sie kein Beispiel anführen, die mockierende Äußerung zum Applewriter-IIc-Patch aus Heft 3/85, S. 35 in Verbindung mit 5/85, S. 68. Tatsache ist doch, daß bis heute (Ende Mai 1985) immer noch der ungepatchte Applewriter IIe an IIc-Besitzer zu deren Frust verkauft wird. So gesehen hätte ich den Daumen noch viel tiefer in die Wunde drücken müssen, damit sich endlich etwas bewegt! us

Erfahrungen mit der Apple-Hotline

Als Einwohner Münchens bin ich weniger durch hohe Telefongebühren der Post behindert, die Apple-Hotline in Anspruch zu nehmen und tue das so alle 1–2 Monate. Ich muß sagen, daß meine bisherigen Erfahrungen wesentlich positiver waren als mit allen ortsansässigen Apple-Händlern zusammen. Während die meisten Händler bei Problemen, die auf fehlende oder unvollständige Systemunterlagen (Apple IIc) zurückzuführen sind, empfehlen, lieber fertige Programme einzusetzen, bei denen diese Probleme nicht auftauchen würden, scheint es in der Ingolstädter Straße auch Fachleute zu geben, die einem weiterhelfen können und wollen! Freilich sind diese Leute auch überfordert, wenn es Apple-Lieferprobleme gibt und diese nach irgendeinem unerfindlichen Lotterieverfahren erledigt werden. Das liegt aber eher an der Behandlung Europas als zweitklassigem Absatzmarkt durch die Amerikaner – siehe Software-Support! Mein letzter Anruf bezog sich auf ein Problem mit dem Horizontal-Tabulator beim Apple-Drucker (Imagewriter), der nicht funktionieren wollte. Einen Tag später bekam ich ein Programmbeispiel zugestellt, mein Problem war gelöst und stellte sich als ein Fehler im Drucker-Manual heraus, in dem ein neuer Funktionscode Ctrl-E noch nicht ausgewiesen war. Danke, Herr Birkl!

Josef Schön, München

II-Plus-Programm auf dem IIc?

Ihre Zeitschrift „Peeker", die insgesamt recht gut gemacht ist, versuche ich regelmäßig zu lesen, wobei ich als computermäßig unbedarfter Apple IIc-Neubesitzer aber gerne gestehe, daß das Niveau Ihrer Darstellungem meine derzeitigen Möglichkeiten bei weitem übersteigt.
Ich möchte den Leserbrief von Herrn Dr. Hickey (Mehr Apple IIc in 5/85) unterstützen und ebenfalls für eine umfangreichere Berücksichtigung des IIc-Typs plädieren. Allerdings sollten Sie hierbei berücksichtigen, daß sich gerade mit diesem Gerät häufig Apple-Neulinge/Einsteiger herumplagen. Momentan ärgere ich mich, daß die angeblich 100%ige Kompatibilität IIc-IIe ja leider nicht zutrifft. Konkret: EZ-DRAW 3.3, das auf unseren Schulrechnern (IIe) problemlos läuft, funktioniert nicht auf meinem IIc.
Mich interessiert, wie man so etwas „lauffähig" macht, wobei ich mir vorstelle, daß einerseits die Problematik von allgemeinerem Interesse ist und andererseits für einen gewieften Programmierer die Aufgabe leicht lösbar sein müßte.

Dipl.Ing.Wolfgang Hannich

aaa electronic gmbh

Neue Preise # 7

IBM-PC-Systemeinheit+Laufwerk	4150,--
IBM deutsche DIN-Tastatur	698,--
IBM-Bildschirm Color	1698,--
IBM-PC/XT-Compatible Computer+Zubehör-	
Mainboard XT f. 16bit+256K-RAM+8 Slots, Leerkarte	149,--
Disk Controller leer (f. 2+2-360KB-Drives)	59,--
Color Graphik Leerkarte	97,--
Parallel Printer Leerkarte	65,--
512K-RAM Leerkarte	97,--
Mainboard XT-16bit+128K-RAM+8Sl. best.+gepr. (bis 256K-aufr.)	797,--
64KB-RAM-Chip-Aufrüstsatz	88,--
Disk Controller gepr. (f. 2+2-360KB-Drives)	269,--
Color Graphikkarte gepr.	447,--
Parallel Printer-Karte gepr. o. Kabel	195,--
512K-RAM-Karte gepr. (0KB)	399,--
Multi I/O Karte (Par.+Ser.+Uhr gepuffert) gepr.	497,--
Funktionstastatur dt. oder ASCII+15er Block	347,--
130W-Netzteil mit eingebautem Ventilator	499,--
360KB-Diskdrive Slimline	498,--
XT-128K-16bit Rechner (bis 256K-aufrüstbar)+360K-Drive +F. Tastatur im IBM-look like Geh.+130W-Netzteil	2.799,--
Software+Manual für IBM+compatible Geräte	
Microsoft flight simulator	145,--
CPM 86	645,--
Lotus 1-2-3	1195,--
Framework	1545,--
D-base III	1575,--
UCSD P-System	1695,--
Symphony	1765,--
APPLE-BUS COMPUTERPLATINEN+PERIPHERIE	
Motherboard durchkontaktet mit Lötstoplack	84,--
M-board II 48K, gesockelt, vollbest.+gepr.	545,--
M-board II 64K+Z80 CPU, gesockelt, v.best.+gepr.	685,--
M-board IIe 64K vollbestückt+gepr.	945,--
APOLLO II **Kunststoff-Leergehäuse**	135,--
APOLLO II-Leergeh. incl. Funkt.-Tast.	275,--
Leerplatine: 16K-RAM, Z80/CPU-, Controller DOS 3.3, Parallel-Druckerkarte, Ser. Int. V24, 80Z/24Z-Karte	
PAL-Modulatorkarte, je	18,95
Entwicklungslochraster-Leerplatine	49,--
16K-Speicher-Karte geprüft	97,--
Z-80 CPU-Karte geprüft	87,--
Controller DOS 3.3-Karte geprüft	88,--
Auto-Controller DOS 3.2/3.3-Karte gepr.	145,--
Parallel-Drucker-Karte+Kabel geprüft	119,--
8bit par. Grappler-Int.+64K-Buffer+Kabel gepr.	575,--
Seriell-Interface-Karte V24 geprüft	175,--
80-Zeichen/24-Zeilen-Karte geprüft	145,--
80Z/24Z-Karte+Softswitch-Schalter Leerplatine	49,--
80Z/24Z-Karte+Softswitch-Schalter geprüft	178,--
IEEE-488 Int. Karte gepr.	375,--
PAL-Modulator/Color-Karte geprüft	175,--
128K Speicher-Leerkarte	89,--
128K-Speicher-Leerkarte+Software	145,--
128K-Speicher-Karte+Software geprüft	439,--
256K-Speicher-Leerk.+4 Spezial ICs+Software	335,--
256K-Speicher-Karte+Software geprüft	898,--
EPROM-Burner (2708/16/32/64) geprüft	215,--
UHF/TV-Modulator universell	27,50
Lüfter anclipsbar (220 V)	87,--
APOLLO // (48K)+UHF-Mod.+Gr/Kl vollbest.+gepr.+F.-T.	875,--
APOLLO // (64K)+Disk II F+Contr.+12"-Monitor	1745,--
APOLLO //A (48K)+Gr/Kl+15er-Tastatur vollbest.+gepr.+F.-T.	1275,--
APOLLO //A (64K)+Disk II F+Contr.+12"-Mon.	1975,--
Aufpreis für frei programmierbare Funktionstastatur plus 4 Cursor-Tasten für APOLLO //A	100,--
APOLLO // ASKF (48K) Sep. Keyboard+Gr/Kl+ 15er-Bl.+Funkt.-Tasten im „IBM-look like" Gehäuse	1275,--
Aufpreis für M-board / 64K+Z80 CPU	115,--
APPLE-DuoDisk, 2 x 143 KB+Controller orig.	2185,--
APPLE-Disk // Laufwerk+Contr.+Kab. DOS 3.3+ProDOS orig.	777,--
APPLE-Disk // (2. Laufwerk) orig.	666,--
Disk //+Contr.+Kabel DOS 3.3 (Siemens) im Geh.	593,--
Disk // (2. Laufwerk, Siemens) im Geh.	498,--
Disk //F+Contr.+Kab. DOS 3.3 (1/2 Höhe) im Geh.	524,--
Disk //F (2. Laufwerk, 1/2 Höhe) i. Geh.	429,--
APPLE ProDOS HB+Disk orig.	69,--
Erphi Duo-Disk 1,2 MByte incl. Netzteil+Contr.	2148,--
Erphi AFDC 2 Controller	288,--
Maus f. Apple // incl. Manual+Software	188,--
APPLESOFT+Turorial+Reference Manual engl.	75,--
APPLE-DOS 3.3 Manual engl.	25,--
APPLE-Pascal Reference+Operating Manual engl.	48,--
APPLE-Pascal USCD 1.1. Disks+Manuals orig.	179,--
APPLE-Pascal Reference Hb. dt.	49,--
APPLE-Pascal Language Hb. dt.	39,--
APPLE-FORTH Manual engl.	69,--
APPLE-Fortran Manual engl.	69,--
CP/M-Software Vol. I+II, Manual engl.	69,--
VIDEX 80Z/Z-Karte Manual engl.	25,--
APPLE // Pascal-Handbuch deutsch (te-wi)	59,--
APPLE //-Anwenderhandbuch deutsch (te-wi)	56,--
6502-Programmier-Handbuch deutsch	45,--
6502-Anwendungen Handbuch deutsch (Rod.Zaks)	38,--
Software+Manual f. APPLE+compatible Geräte	
apple how to, apple soft tool kit, cp/m basic compiler, cbasic, dbase // ver. 2.4, wordstar+mailmerge+spellstar mit installprogramm, apple logo, datastar, calcstar, pfs pfs, pfs graph, pfs re+port, apple fortran, cp/m c language, spitfire simulator, c-programming language, flight simulator, visicalc, visischedule, visitrend, multiplan, supersort, microsoft cobol 80, microsoft fortran 80, dgraph+dbase utility.	
APOLLO //e (64 K) PAL+15er Block * NEU * NEU * NEU *	1595,--
APOLLO //e 128K+80Z/24Z-Karte	1825,--
APPLE //e (64K) PAL	2345,--
APPLE //e (64K)+Disk II F+Contr+12"-Monitor	3125,--
80Z/24Z+Softswitch+64K-RAM-Karte geprüft	249,--
APPLE //c (128K)	2665,--
Z80 Karte gepr. f. APPLE //c	489,--
Macintosh System (128K)	5945,--
Macintosh System (512K)	6987,--
EPSON RX 80, 8bit/parallel	869,--
EPSON RX 80F/T, 8bit/parallel	1019,--
EPSON RX 100, 8bit/parallel	1289,--
EPSON FX 80F/T, 8bit/parallel	1329,--
EPSON FX 100, 8bit/parallel	1739,--
EPSON Traktoraufsatz für FX 80	129,50
APPLE/EPSON-Drucker Graphic-Interface+Kabel	295,--
MX 80/82, FX/RX 80 Spezialfarbband-Kassette	15,95
BROTHER-Typenraddr. HR 15XL 8bit/par. neustes Mod.	1387,--
STAR SG10, 8bit/par.	1099,--
Zenith ZVM 123 gr. 12" oder **Zenith ZVM 122** or. 12" 15 MHz	277,--
SANYO DM 2212 12" orange 15 MHz	295,--
APPLE // Video-Mon. 12" 20 MHz, entspiegelt orig.	398,--
5¼" 1D-Disk-Soft-verstärkt, Neutral, 1. Wahl 10er Pack	32,50
5¼" 1D-Disk-Soft-verstärkt, Neutral, 1. Wahl 100er Pack	297,--
5¼" 1D-Disk-Soft-verstärkt, Neutral, 1. Wahl 1000er Pack	2799,--
5¼" 2D-Disk-Soft-verstärkt 10er Pack	47,50
5¼" 2D-Disk-Soft-verstärkt 100er Pack	455,--
Data Magnetics-Disketten-verstärkt 1. Wahl	
5¼" Softsektoriert 10er Pack	35,50
5¼" Softsektoriert 100er Pack	325,--
5¼" 1D-Softsektoriert 10er Pack	39,50
5¼" 1D-Softsektoriert 100er Pack	379,--
3M-Scotch-Disketten – verstärkt in Box+Aufkl. 1. Wahl	
3½" Microdisketten 10er Pack	149,50
5¼" 1D-Disk-Soft- o. Hardsektoriert 10er Pack	48,90
5¼" 1D-Disk-Soft- o. Hardsektoriert 100er Pack	478,--
5¼" 2D-Disk-Soft- o. Hardsektoriert 10er Pack	69,90
5¼" 2D-Disk-Soft- o. Hardsektoriert 100er Pack	668,--
5¼" 1/96 Softsektoriert 10er Pack	81,50
5¼" 1/96 Softsektoriert 100er Pack	769,--
5¼" 2/96 Softsektoriert 10er Pack	88,90
5¼" 2/96 Softsektoriert 100er Pack	849,--
8" Soft- o. Hardsektoriert 10er Pack	67,--
8" Soft- o. Hardsektoriert 100er Pack	649,--
8" 1D-Disk-Softsektoriert 10er Pack	86,--
8" 1D-Disk-Softsektoriert 100er Pack	829,--
8" 2D-Disk-Softsektoriert 10er Pack	99,--
8" 2D-Disk-Softsektoriert 100er Pack	959,--
3M-Scotch 5¼" Flüssig-Reinigungsset (2 Disk.)	48,95
BASF Disketten – verstärkt in Box+Aufkleber 1. Wahl	
5¼" 8" Soft- o. Hardsektoriert 10er Pack	45,--
5¼" o. 8" Soft- o. Hardsektoriert 100er Pack	439,--
5¼" o. 8" 1D-Softsektoriert 10er Pack	48,50
5¼" o. 8" 1D-Softsektoriert 100er Pack	479,--
5¼" o. 8" 2D-Softsektoriert 10er Pack	71,--
5¼" o. 8" 2D-Softsektoriert 100er Pack	698,--
5¼" 1/96 Softsektoriert 10er Pack	67,50
5¼" 1/96 Softsektoriert 100er Pack	659,--
5¼" 2/96 Softsektoriert 10er Pack	81,50
5¼" 2/96 Softsektoriert 100er Pack	788,--
5¼" 2/96 High Density 1.2 MByte 10er Pack.	139,--
5¼" Reinigungsdiskette einseitig	48,50
5¼" Disketten-Archivbox für 10 Disks	5,75
5¼" Disketten-Archivbox 10er Pack	58,95
5¼" Disketten-Archivbox für 15 Disks	19,95
5¼" Disk-Karteikasten Kunststoff (ca. 80 Disk)	29,95
5¼" Disk-Karteikasten Kunststoff (ca. 80 Disk)	49,95
5¼" 3M-Disk-K.K. abschließbar (ca. 50 Disk)	45,95
5¼" Disk-K.K. Rauchglas, abschließb. (ca 80 Disk)	67,95
2000 Bl. Tabpapier (24 cm x 12" einf.) weiß oder grün/weiß perf.	47,--
4000 Et.-Aufkl. (107 x 36 mm Doppelr. 240 x 12" perf. Trägerp.)	66,50
16000 Etiketten-Aufkleber w.o.	249,--
4000 Etik.-Aufkl. einreih. (107 x 36 cm auf 12,5 x 12" TrägerPap)	65,--

OSZILLOSCOPE HAMEG ab Lager
Bei Vorauszahlung frei Empfangsstation unversichert in der BRD, ausgenommen Papier und Etiketten, sonst N.N.+V.S. ab DM 30,--

aaa electronic gmbh
Telex: 07 72 642 aaa-d
Habsburgerstraße 134
7800 FREIBURG, Tel. (07 61) 27 68 64
Bauelemente – Bausätze – µP's
Meßgeräte – Zubehör – Fachliteratur
Fachgeschäft für Elektronik + Mikrocomputer

HCV

Der Einsteiger
Apple IIe 64 KB
Apple Monitor II
Apple Disk II
m. Contr.
DM 3998,–

Der Tragbare
Apple IIc 128 KB
80 Zeichen
Disk integr.
DM 2998,–

Der Vielseitige
Apple Macintosh
128 KB
incl. MacPaint/
MacWrite
Tastatur und Maus
DM 6776,–

Der Schnelle
Apple „Imagewriter"
180 Z/sec. – Graphik
DM 1598,–

Die Zuverlässige
80 Zeichen/64 KRAM
m. Pseudo-Floppy-
Software u. ausführlichem Handbuch
DM 298,–

Die Flache
Slimline Disk
m. Apple Contr.
Diskette u. deutschem
Handbuch
DM 698,–

Der Schöne
Brother HR 15 XL
Typenraddrucker,
15 Z/sec.
DM 1398,–

Der Brillante
Taxan Monitor 12" grün
22 MHZ Auflösung
DM 398,–
RGB-Monitore auf Anfrage

Bei Abnahme von größeren
Stückzahlen noch
günstigere Preise
möglich.

Preise incl. MwSt.
1 Jahr Garantie
Rückgaberecht innerhalb
von 8 Tagen
Kostenloses Info anfordern

Hamburger
Computer
Versand
☎ 040-7928226
Postfach 61 01 25
2000 Hamburg 61

Wenn ein Programm erstens die Sprachkarte = 16K-Erweiterung und zweitens die 80-Zeichenkarte *nicht* benutzen, dann können Sie alle älteren Programme, die ursprünglich für den Apple II Plus gedacht waren, problemlos verwenden, wenn Sie *FPBASIC* in die Sprachkarte laden:

1. DOS 3.3 booten
2. Ctrl-Reset drücken (Damit geht man sicher, daß die 16K-Erweiterung schreibfähig gemacht wird.)
3. BLOAD FPBASIC, A$D000
4. CALL -151
5. C083 C083
6. Ctrl-C
7. Ab jetzt kein Ctrl-Reset und kein PR#3 mehr!
8. Altprogramm mit RUN XXX oder BRUN YYY starten.

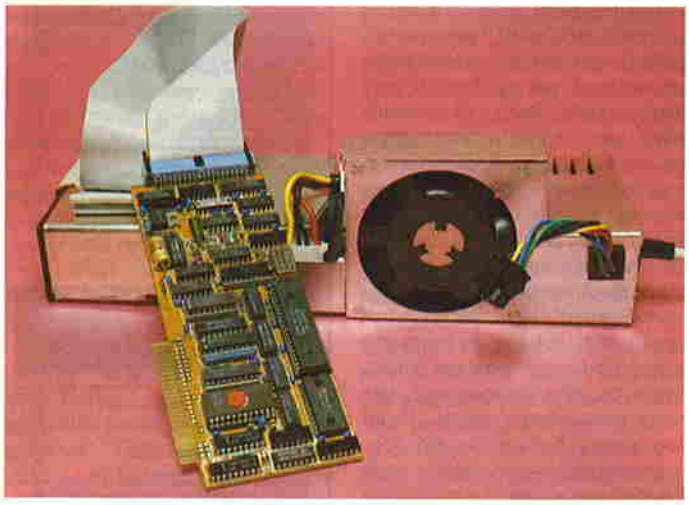
Es versteht sich, daß dieses Verfahren nur dann funktioniert, wenn man ungeschützte Programme verwendet, die *nicht* mit PR#6 gebootet werden müssen. Ferner beachte man, daß man das 50 (*fünfzig*) Sektoren umfassende FPBASIC von der *alten* System-Master-Diskette laden muß (also nicht ein um das Monitor-ROM verkürztes FPBASIC!). Schließlich sei darauf hingewiesen, daß sich IIe-Programm, die nicht auf dem IIc laufen, auch dann nicht simulieren lassen, wenn man den IIe-ROM-Inhalt in die Sprachkarte lädt. Zu diesem Zweck müßten nämlich zahlreiche Adresse geändert werden.

us

MEGACORE

Festplatte mit 10 MBytes

getestet von H. Grumser



Die Firma Compu Shack vertreibt eine 10-MByte-Festplatte englischer Herkunft mit dem Controller MEGABOARD, der von der Frank & Britting Elektronik Entwicklungs-GmbH entwickelt (und der Peeker-Redaktion auch von dort zu Testzwecken zugesandt) wurde. Die Harddisk wird an Stelle des Apple-Netzteils eingesetzt und übernimmt dann die Spannungsversorgung des Computers. Der Lieferumfang umfaßt neben einem deutschen Handbuch auch die entsprechende Software (5,25-Zoll-Disketten), die zur Initialisierung und Konfigurierung der Platte erforderlich ist.

Inbetriebnahme

Laut Angabe von Compu Shack kann der Einbau in drei Minuten vollzogen werden. Wer das ca. DM 5000.– teure Gerät und seinen nicht minder wertvollen Apple nicht aufs Spiel setzten möchte, muß mit einer halben Stunde rechnen. (Herr Grumser ist Physiker, also kein „blutiger Laie". Anm.d.Red.)

Die erste Inbetriebnahme läßt einige Sekunden an der vollbrachten Arbeit zweifeln, da der gewohnte Piepston zunächst ausbleibt. (Vor der Versorgung des Rechners wird das Laufwerk hochgefahren.) Die Präsenz der Harddisk kann stets durch das Laufwerksgeräusch vernommen werden. Wen dieses Summen stört, kann die Festplatte „auslagern" und unter dem Schreibtisch deponieren.

Nach dem Booten der mitgelieferten Diskette in einem Floppy-Disk-Laufwerk kann die Harddisk initialisiert werden, wobei der Inhalt der Diskette auf die Festplatte kopiert wird. Somit steht in jedem Fall ein DOS-Volume zur Verfügung, das beim nächsten Kaltstart gebootet wird.

Betriebssysteme

Für die künftige Arbeit mit der Festplatte sollte man nach dem Initialisieren Bereiche für die verschiedenen Betriebssysteme einrichten. Eine spätere Änderung dieser Organisation ist mit einem erheblichen Datentransfer auf der Platte verbunden und kann zum Verlust von Einträgen führen, da die vier Bereiche stets aufeinanderfolgend angeordnet sind. Diese Konfigurierung erfolgt durch Aufruf der einzelnen Betriebssystem-Manager aus dem Hauptmenü:

DOS 3.3 – Da unter DOS 3.3 keine größeren Datenmengen verwaltet werden können, wird dieser Bereich in einzelne Volumes unterteilt, die dann durch Volume- und Drive-Nummer angesprochen werden. Die somit mögliche Simulierung von 2 * 255 Floppy-Disk-Laufwerken (je 140 KBytes) erlaubt die Benutzung der kompletten Festplatte unter DOS 3.3.

CP/M – Unter CP/M (Version 2.2 56K) können 6 Volumes (A–F) mit unterschiedlicher Größe verwaltet werden (128–7680K pro Volume in 128K-Schritten). Um weiterhin auf „normale" Disketten zugreifen zu können, besteht die Möglichkeit, ein oder mehrere Volumes zu desaktivieren, d. h. der Zugriff erfolgt dann auf das entsprechende Diskettenlaufwerk und nicht mehr auf die Festplatte.

UCSD-Pascal – Der Zugriff unter UCSD-Pascal (Apple-Pascal 1.1) muß wie oben eingestellt werden. An Stelle der Volumes treten hier die Units 4, 5, 9, 10, 11 und 12, die mit beliebiger Länge konfiguriert werden können.

ProDOS – Die Einrichtung eines ProDOS-Bereichs (ProDOS Version 1.0.1) ist wegen der Konzeption dieses Betriebssystems für größere Massenspeicher sehr einfach. Es genügt die Eingabe der gewünschten Größe, die dann als ein Volume verwaltet wird.

Welches Betriebssystem beim Kaltstart gebootet wird, kann mit Hilfe des Boot-Managers gewählt werden. Die mit einer der mitgelieferten Disketten gepatchten Systeme müssen jedoch vorher auf die Harddisk kopiert werden. Spezielle Treiberprogramme erlauben die Umschaltung von einem Betriebssystem zum nächsten ohne Kaltstart, wobei stets der Weg über das beim Initialisieren angelegte DOS-3.3-Volume beschritten wird.

Die entsprechende Patch-Software für andere Betriebssystem-Versionen ist in Vorbereitung und kann individuell erfragt werden.

Handbuch

Das spiralgebundene Handbuch (branchenübliches Schönschreibdrucker-Printout) beschreibt auf ca. 50 Seiten sehr ausführlich die Anwendung der mitgelieferten Programme und die Inbetriebnahme der Harddisk bezüglich Formatierung und Konfigurierung. Auf Schwierigkeiten bei der Umorganisierung der Platte wird ebenso eingegangen wie auf die Erstellung von Sicherungskopien.

Als besonders hilfreich für Assembler- und System-Programmierer erweist sich die Beschreibung des WD1010-Prozessors von Western Digital als Controllerchip und eine Erläuterung zur Programmierung des Sektorzugriffs auf Systemebene. Die transparente Dokumentierung erinnert an die „gute alte Apple-Zeit".

Übertragungsrate
Die Übertragungsrate kann nur auf Systemebene exakt ermittelt werden. Eine weniger zeitaufwendige Lösung ist die Verwendung eines Benchmark-Tests. Die zwei kurzen Programme **WRITE.TEST** und **READ.TEST** ergeben unter ProDOS eine Übertragungsrate von ca. 4000 Bytes/s beim Schreiben und ca. 15400 Bytes/s beim Lesen. Dies sind reine Brutto-Werte, da die systeminterne Verwaltungszeit (insbesondere die Bearbeitungszeit des Applesoft-Programms) mit gemessen wird. Ein ähnlicher Test bei der Profile (in „Apple ProDOS für Aufsteiger", S.18, Hüthig Verlag) ergab etwa die gleichen Werte.

Anmerkung zur MEGACORE/MEGABOARD unter ProDOS: Die Harddisk wird unabhängig vom Controller-Slot unter „S7, D1" angesprochen (gilt auch für DOS 3.3). Die RAM-Disk bleibt erhalten. Der Gerätetreiber wird über $BF1E/$BF1F angesprungen und befindet sich auf der Controller-Karte ab $C7D0.

Fazit
Die Kombination aus MEGACORE und MEGABOARD bietet einen kompatiblen Übergang auf Massenspeicher höherer Kapazität. Die Übertragung erfordert nur wenige Eingriffe in bereits vorhandene Programme, da die Anpassung an die verschiedenen Betriebssysteme unter Einhaltung der Spielregeln vorgenommen wurde. Für Anwender mit erhöhten Datenmengen ist dieses System daher empfehlenswert.

```
WRITE.TEST (255s)

10 PRINT CHR$ (4)"OPEN XXX"
20 PRINT CHR$ (4)"WRITE XXX"
30 X$ = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
40 FOR X = 1 TO 5000: PRINT X$: NEXT
50 PRINT CHR$ (4)"CLOSE"
60 PRINT CHR$ (7)

READ.TEST (65s)

10 FOR Y = 0 TO 99: X = X * 1000
20 PRINT CHR$ (4)"BLOAD XXX,TTXT,A10000,L10000,B"X
30 NEXT
40 PRINT CHR$ (4)"CLOSE"
50 PRINT CHR$ (7)
```

Star Delta-10
und Grafstar-Interface

getestet von Karl-Walter Bott

Gerade bei Matrixdruckern ist die Angebotsvielfalt der einzelnen Hersteller für den Käufer kaum noch zu überblicken.
Vor geraumer Zeit konnte man Matrixdrucker noch in zwei Gruppen einteilen. Zur ersten Gruppe gehörten die preiswerten, langsamen Drucker mit mäßigen Grafikfähigkeiten und minderer Ausgabequalität. Zur zweiten Gruppe zählten die Produkte, die durch ein sehr gutes Schriftbild, hohe Grafikauflösung und Schriftqualität überzeugten, aber weniger durch einen günstigen Preis. Mit dem **Star Delta-10** erwirbt man einen Matrixdrucker, der mehr kann, als nur Listings zu Papier bringen, und dabei ein günstiges Preis-Leistungsverhältnis aufweist.

Das Aufstellen des Druckers bereitet keine Schwierigkeiten, wenn man die mitgelieferte deutsche Bedienungsanleitung zur Hand nimmt.
Auf der Rückseite des Gerätes befinden sich die zwei Anschlüsse der Schnittstellen zum Computer. Der Delta-10 ist serienmäßig mit einer Parallelschnittstelle und einem seriellen Interface RS232C ausgestattet. Zur Standardausrüstung gehört auch ein 8K großer Druckerpuffer. Ist die Verbindung zum Computer hergestellt, müssen noch die drei DIP-Schalter (DIP = Dual Inline Package) gesetzt werden, von denen nur einer auf der Rückseite frei zugänglich ist. Die beiden anderen können nur nach dem Abnehmen der Gehäuseabdeckung bedient werden. Mit den DIP-Schaltern können Parameter voreingestellt werden, mit denen der Drucker beim Einschalten initialisiert wird: Zeichensatz, Formularlänge, Zeichenabstand, Zeilenvorschub, Auswahl der Schnittstelle sowie Betriebsart und Baud-Rate bei seriellem Betrieb.
Neben dem normalen ASCII-Zeichensatz gibt es 88 internationale Zeichensätze (bzw. deren landesspezifische Sonderzeichen wie ß, Ä, Ö, Ü usw.), 96 Kursiv-Zeichen, 32 Blockgrafik-Zeichen, 64 Spezial-Zeichen sowie 2 mal 96 frei programmierbare Zeichen.

Normale Schrift
Überall ist Platz
für manchen klugen Satz.

gedehnte Schrift

Überall ist Platz
für manchen klugen Satz.

Kursivschrift
Überall ist Platz
für manchen klugen Satz.

Elite Schrift
Überall ist Platz
für manchen klugen Satz.

Komprimierte Schrift
Überall ist Platz
für manchen klugen Satz.

Doppeldruck
Überall ist Platz
für manchen klugen Satz.

Hervorgehobener Druck
Überall ist Platz
für manchen klugen Satz.

Unterstreichen
Überall ist Platz
für manchen klugen Satz.

Hochgestellt
$Y=Ax^3$

Tiefgestellt
H_2O

Die Vielzahl der Druckmodi bietet jedem Benutzer etwas, vor allem, weil diese untereinander gemischt werden können. Alle Schriftarten und Druckmodi sind vom Programm her auszuwählen. Die Steuerzeichen sind weitgehend Epson-kompatibel. Der Zeilenvorschub ist ebenfalls programmierbar. Die kleinste Schrittweite ist n/144 Zoll.

Das Drucken erfolgt bidirektional mit einer Geschwindigkeit von maximal 160 Zeichen pro Sekunde. Die maximale Anzahl von Zeichen pro Druckzeile beträgt 132. Die Druckgeschwindigkeit zusammen mit dem Druckerpuffer dürfte auch Benutzer zufrieden stellen, die überdurchschnittlich viel zu drukken haben.

Um Grafiken auf das Papier zu bringen, gibt es neben der Möglichkeit, Zeichenmuster selbst zu definieren, den Bit-Image-Grafikmodus. Die maximale Auflösung durch die Einzelnadelansteuerung beträgt 1920 Punkte und ist auch auf dem Papier durchaus überzeugend.

Als weitere Besonderheiten des Druckers sind noch Macro-Instruktionen zu nennen, mit deren Hilfe Steuersequenzen gespeichert werden können, sowie der eingebaute Selbsttest.

Der Delta-10 verarbeitet fast alle gängigen Papierarten und Formularabmessungen: Einzelblatt, Rollenpapier und Endlospapier. Als Farbband dient ein normales Schreibmaschinen-Farbband.

Nach so vielen positiven Aspekten sollten die Nachteile des Druckers nicht verschwiegen werden. Zunächst ist hier die Anordnung des Traktors zu nennen. Der Traktor wird oberhalb des Druckkopfes auf den Drucker aufgesetzt. Der Nachteil dieser Anordnung liegt in der mangelnden Ausnutzung des Endlospapiers. Will man den Druck am Anfang einer neuen Seite (Perforation) beginnen, wird jedesmal das erste Blatt verschenkt, da der Druckkopf unterhalb des Traktors angeordnet ist. Bei der Verarbeitung von Einzelblattpapier muß der Traktor abgenommen werden. Das Einziehen des Einzelblattes gestaltet sich etwas schwierig, zumindest, wenn das Blatt exakt horizontal und vertikal ausgerichtet sein soll, da keine Markierung am Gehäuse angebracht ist. Eine aufgeklebte Plastikschiene, mit einer Längeneinteilung am Gehäuse oberhalb der Walze, würde hier Abhilfe schaffen. Um das Blatt vertikal zu justieren, muß das Blatt so weit eingezogen werden, daß die Enden parallel ausgerichtet werden können. Danach kann man dann das Blatt in die erste Druckzeile zurückdrehen. Dies empfiehlt sich mit Rücksicht auf die Druckermechanik nur, wenn der Drucker ausgeschaltet ist. Also bei z.B. 20 Serienbriefe 20mal Ein- und Ausschalten, was auch nicht gerade die Lebensdauer der Elektonik erhöht.

Ansonsten machen Mechanik und Elektronik einen soliden Eindruck.

Zur Kommunikation zwischen Apple-Computern und Star-Druckern wird das Parallel-Interface **Grafstar** angeboten. Das Interface ermöglicht eine Hardcopy von den Grafik- und Textseiten, wobei auch eine Vergrößerung, Drehung und Invertieren der Grafik-Hardcopy möglich ist. Die Steuerzeichen sind weitgehend Epson-kompatibel. Eine Besonderheit ist die Möglichkeit, die Ausgabe über eigene Assembler-Routinen umzuleiten, die entweder ab $0300 oder $D000 der Language-Card beginnen können.
Wenn das Interface in Slot 1 installiert ist, treten keine Probleme auf. Die Firmware arbeitet unter CP/M, UCSD, DOS und ProDOS einwandfrei. Mit den testweise ausprobierten Programmen Appleworks, Applewriter und Wordstar 3.3 funktionierte das Grafstar-Interface ohne Probleme.

aaa electronic gmbh, Freiburg 75
ccp-datentechnik, Hamburg 15
Copy-Team GmbH, Erlangen 68
CP/D oHG, Düsseldorf 15
U. Dobbertin, Brühl 51
D.O.S. Computersysteme, Schwäbisch Hall 32
Forth-Systeme A. Flesch, Titisee-Neustadt 15
Ingenieurbüro Fricke, Berlin 73
Hamburger Computer Versand, Hamburg 75
IBS Computertechnik, Bielefeld . . . . . U 2
Interkom electronic, Isernhagen 73
Intus, Waldshut-Tiengen 49
Klaus Jeschke, Kelkheim 74
E.-W. Meyer, Frohnhausen 49
Micromint Computer GmbH, Erkrath 15
U. Mohwinkel Electronic, Leverkusen 49
Pandabooks, Berlin 11
Pandasoft, Berlin 13
r + r electronic, Heidelberg 61
R.v. Decker's Verlag, Heidelberg 56
Röckrath Microcomputer, Aachen 49
Springmann Computer GmbH, Hannover 74
Summagraphics, München 52
te-wi-Verlag GmbH, München . . . . . U 4
Tombstone-Micro, Berlin 51
Ueding electronics, Menden 22

Einem Teil dieser Ausgabe liegt ein Prospekt der Firma Interdata GmbH, Singen bei.
Wir bitten unsere Leser um Beachtung.

. . . in eigener Sache:

die Nachfrage nach dem vergriffenen Heft 1/84 ist groß. Ab sofort können Sie eine Heftkopie direkt beim Verlag bestellen.

Preis für das Inland:
DM 10,– inkl. Versandkosten

für das Ausland:
DM 12,– inkl. Versandkosten
eventuelle Luftpostzuschläge zzgl.

Der nächste Peeker
Heft 8/1985
erscheint am
22. 7. 1985

Hüthig-FACHBUCH-TIP

Band 2
erscheint Anfang Juli
ca. DM 30,—

„Apple ProDOS für Aufsteiger" ist der Nachfolgeband zu „Apple DOS 3.3 —Tips und Tricks". Applesoft-Programmierer, die unter DOS 3.3 gearbeitet haben, werden sich schnell an ProDOS gewöhnen, da ProDOS und DOS 3.3 in dieser Hinsicht weitgehend kompatibel sind. Dagegen müssen Assembler-Programmierer völlig umdenken. Deshalb liegt das Schwergewicht dieses Nachfolgebandes auf der Assemblerprogrammierung und der minutiösen Darstellung der ProDOS-internen Systemadressen, die jedoch auch für Applesoft-Programmierer von großer Bedeutung sind.
Im ersten Teil wird zunächst ein allgemeiner Überblick über das neue „Professional Disk Operation System" gegeben. Im Anschluß daran folgt eine Gegenüberstellung der Geschwindigkeit des Diskettenzugriffs. Dann wird die interne Speicherorganisation detailliert beschrieben (Boot-Vorgang, Zero-Page, ProDOS-Vektoren, Basic-System-Puffer, Basic-System-Global-Page, Basic-Command-Handler, I/O-Vektoren, ProDOS-Global-Page, Language-Card-Organisation, Interrupt, Disk-Driver, Reboot-Programm usw.).

Ebenso ausführlich wird die externe Speicherorganisation geschildert (Spuren, Sektoren, Blocks, Directory-Struktur, Volume Bit Map, Dateistrukturen usw.). Schließlich wird das MLI (Machine Language Interface) mit zahlreichen praktischen Anwendungsbeispielen erläutert. Insgesamt enthält ProDOS-Buch ca. 70 Seiten mit eigens für dieses Werk entwickelten Programmen.
Im zweiten Teil werden die Basic-System-Befehle für Applesoft-Programmierer systematisch erläutert. Allerdings wird die Kenntnis von „Apple DOS 3.3" vorausgesetzt. „ProDOS für Aufsteiger" ist deshalb nicht nur für Assembler —, sondern auch für fortgeschrittene Appelsoft-Programmierer ein unentbehrliches Nachschlage- und Handbuch für die Programmierpraxis.

Apple ProDOS für Aufsteiger

Mit ausführlichen Programmbeispielen

von Ulrich Stiehl

Band 1: 2. geänderte Auflage 1985, 208 S., kart., DM 28,—

BESTELLCOUPON

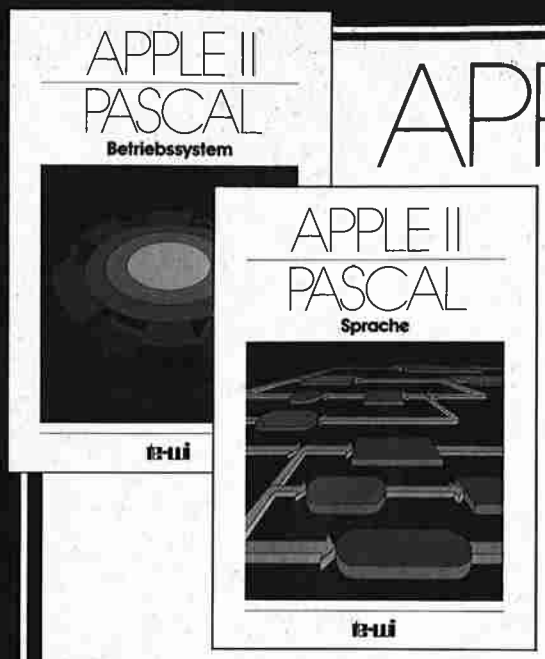
Bitte ausfüllen und an Hüthig Vertriebsservice, Postfach 102869 · 6900 Heidelberg schicken.

Buchtitel _____

Name _____ Ort _____

Straße _____

Unterschrift _____



APPLE II PASCAL

Betriebssystem und Sprache

Erstes deutsches Referenzwerk sämtlicher Befehle und Systemroutinen von Apple II Pascal – mit Addendum einschließlich Version Pascal 1.2!

Gültig für Apple II, II Plus, IIe einschließlich der 128K/80 Zeichen-Konfiguration.

Betriebssystem kommentiert ausführlich und in Deutsch Funktion und Benutzung der fast 60 Systemroutinen des Apple II Pascal Betriebssystems.

Sprache ist das vollständige, deutsche Referenzwerk der „Apple Pascal"-Programmiersprache mit u. a. Informationen über professionelle Pascal-Programmierung, Turtlegraphics, Programmbibliothek etc.

In Vorbereitung: Addendum Pascal 1.2, ein Zusatz zum Buch „Betriebssystem" für 1.2-Benutzer in Deutsch.

„Nach Unterlagen von Apple Deutschland hergestellt"

Apple II Betriebssystem, 272 Seiten, DM 49,–

Apple II Sprache, 216 Seiten, DM 39,–

Pascal 1.2 Addendum, 112 Seiten, DM 36,–

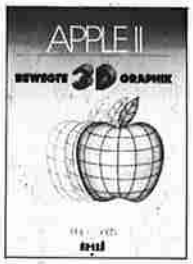
te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

Weiterführende Literatur...

Das APPLE II – Handbuch NEU
(L. Poole)
Erst mit Hilfe dieses Leitfadens werden Sie Ihren Apple II erfolgreich einsetzen, denn Text und Bildmaterial gehen weit über das hinaus, was herstellerseitig an Literatur angeboten wird.
Neu überarbeitet und jetzt um die spezifischen Eigenheiten der Modelle **IIe** und **IIc** erweitert. 500 Seiten, Softcover, DM 66,–

LOGO – Jeder kann programmieren
(Daniel Watt)
Buch des Jahres in den USA. Für die Computer APPLE II, C-64, IBM PC, ATARI bis 520 ST., TI-99 und CPC 464/664. Hochwertiges Textbuch für Logo-Kurse für zu Hause und im Lehrbereich.
384 Seiten, A4, DM 59,–

APPLE II PASCAL – Eine praktische Anleitung
(A. Luehrmann, H. Peckham)
Unentbehrlich für alle, die die Programmiersprache PASCAL lernen wollen und Zugang zu einem Apple-Computer haben.
544 Seiten, Softcover, DM 59,–



APPLE II – Bewegte 3D-Graphik
(Phil Cohen)
Selbstentworfene Graphiken und Diagramme – animiert oder als Standbilder – eben oder räumlich: alle erforderlichen BASIC-Programme mit Erklärung finden Sie in diesem Buch.
200 Seiten, Softcover, DM 49,–

Computer für Kinder
(Sally Greenwood Larson)
Ein Buch für Kinder, ihre Lehrer und Eltern.
„Computer für Kinder" richtet sich an Kinder im Alter von 8 bis 13 Jahren, für deren Interesse an Computern dieses Buch bewußt geschrieben wurde.
Unterhaltsam und leicht verständlich.
A4 quer, Fadenheftung, DM 29.80

Apple Maschinensprache
Für BASIC-Programmierer der einfachste Zugang zur Muttersprache des Apple. Wesentlich schnellere Maschinenprogramme, direkte Manipulation des Mikroprozessors 6502 im Apple – als Brücke dorthin benötigt dieses Buch nur die drei BASIC-Befehle, POKE, CALL, PEEK. D. Inman/K. Inman. DM 49,–

NEU: REPARATURANLEITUNG COMPUTER: APPLE II, II PLUS DM 29,80

tm 4298

Noch im Programm:
6502 – Programmieren in Assembler DM 59,–
VisiCalc, 50 Programme auf Diskette, DM 79,–

In Vorbereitung:
Macintosh Programmier-Handbuch DM 59,–